The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor

Oscar Karnalim[1] and Mewati Ayub[2]
Faculty of Information Technology, Maranatha Christian University, Bandung 40164, Indonesia
Email: [1]oscar.karnalim@it.maranatha.edu, [2]mewati.ayub@it.maranatha.edu

***Abstract*—Program Visualization (PV) is an educational tool frequently used to assist users for understanding a program flow. However, despite its clear benefits, PV cannot be incorporated easily on Introductory Programming course. Several key properties such as student characteristics and behavior should be considered beforehand. This paper is intended to provide an empirical review about the impact of PV toward students of Introductory Programming course. For our case study, PythonTutor is selected as a sample of PVs due to its accessibility. It can be accessed anywhere and anytime through a web browser. Three conclusions are obtained based on our evaluation on data collected from a survey. Firstly, PV is quite effective to assist students for conducting several programming sub-tasks. Secondly, PV, at some extent, may help students to learn advanced topics on Introductory Programming course. Finally, despite the fact that several features should be incorporated to enhance understanding of students, PV is beneficial for learning Introductory Programming course, especially when it is frequently used.**

***Index Terms*—Program Visualization, Introductory Programming course, Empirical Evaluation, Educational Technology**

I. Introduction

PROGRAMMING is a core topic in information technology program study. However, not all students can master it properly due to its complexity [1]. Several students need more time to learn it while the others need more stresses on some program parts to understand it. Thus, several educational tools, which may assist students to understand programming further, are developed. These tools target various aspects of programming such as abstraction knowledge [2], program flow [3], and program characteristics [4, 5]. Among numerous educational tools, Program Visualization (PV) is an educational tool focusing on the second aspect, program flow [6]. Using the tool, users can understand how their program behaves and learn why such behavior occurs.

Currently, there are various PVs developed to assist users to understand programming. Jeliot 3 [7], JIVE [8], VILLE [9], and PythonTutor [3] are four prominent examples in this category. Even though these PVs share similar major goal, they are still different in the term of their features. For instance, VILLE [9] incorporated source code in a parallel view as its unique feature so that they could enhance student understanding in the term of syntax correlation.

Despite the fact that there are many available PVs which may help students to understand programming further, incorporating the tool into Introductory Programming course is not a trivial task. Several key properties such as student characteristics and behavior should be taken into account. Thus, this paper is intended to provide an empirical review about the impact of PV in Introductory Programming course. This review is expected to help lecturers to consider whether PV should be incorporated on their Introductory Programming course or not. In general, there are three aspects evaluated in this work. These aspects are: 1) PV impacts on assisting students to conduct several programming sub-tasks such as understanding the concept of variable; 2) PV impacts on assisting students to understand advanced topics on Introductory Programming course; and 3) Students' feedbacks about the PV features and its usage. These aspects are evaluated based on two-fold which are questionnaire survey and quiz results. Yet, to get more accurate result, both evaluation schemes are conducted after the students have been introduced with PV for six weeks. In this work, we select PythonTutor as our sample PV. This tool is preferred to other tools due to its accessibility. It can be accessed anywhere and anytime since it is designed as a web-based application.

Received: 27 Jun. 2017; accepted: 17 Oct. 2017; available online: 28 Nov. 2017.

Cite this article as: O. Karnalim and M. Ayub "The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor", CommIT (Communication & Information Technology) Journal 11(2), 67–76, 2017.

II. Literature Review

According to several works [10–13], learning programming has been proven to be a highly non-trivial task for students. Students are not only required to master considerably high computational thinking but also well-developed knowledge about programming language. Based on that reason, many educational tools have been developed to simplify student learning process, especially for novice students. These tools target various programming aspects. It may target abstraction knowledge [2], program flow [3], or program characteristics [4, 5]. However, since program characteristics are considered as supplementary knowledge, the target is rarely used as the main objective of educational tools.

Based on the fact that illustration may help students to recall explained information [14], most educational tools are focused on visualizing the learning material. The tools focused on visualizing abstraction knowledge is frequently referred as Algorithm Visualization (AV) [15], whereas the others that focus on visualizing program flow are referred as Program Visualization (PV) [6]. AV is an educational tool which visualizes and animates several aspects from a given algorithm based on its working mechanism [16]. VisuAlgo [2, 17, 18], AlgoViz [19], AP-ASD1 [20], AP-SA [21], and AP-BB [22] are several examples in this category. Some of them are developed as a web-based application to enable easy access, whereas the others are developed as a desktop-based application to enable direct use without relying on Internet connection. On the contrary, PV is an educational tool which visualizes and animates several program aspects from a given source code based on its runtime execution [6]. This tool is frequently used as a supplementary tool to Introductory Programming course [3, 6, 13, 23]. Several examples of the tools are Jeliot 3 [7], JIVE [8], VILLE [9], and PythonTutor [3]. Among these tools, PythonTutor is the only tool designed as a web application.

Jeliot 3 [7] is a program visualization tool aiming at novices for learning Java programming. As the execution advances step-by-step, all variables and function calls are visualized to improve user understanding further about the given program. According to several comprehensive evaluations [13, 24, 25], Jeliot 3 can enhance user understanding in learning Java program.

JIVE [8] is quite similar to Jeliot 3 and only differs on several key features such as object structure and calling method sequence visualization. However, the main focus of JIVE is inclined to visualize program in an interactive manner, and it is not specifically focused on learning programming. According to their evaluation, JIVE can be a practical tool for visualizing and debugging program.

VILLE [9] is a program visualization tool which is designed as language-independent as possible. Unlike Jeliot 3 and JIVE, VILLE can incorporate any programming languages as long as their syntaxes have one-to-one correspondence with Java and C++ syntaxes. Furthermore, VILLE is also featured with parallel view where users can learn syntax translation between two programming languages. As the execution advances step-by-step, VILLE highlights the executed syntaxes which refer to similar semantic from both source codes. According to several comprehensive evaluations [23, 26], VILLE can be considerably effective for learning programming.

PythonTutor is a web-based program visualization tool which is initially focused on visualizing Python codes for Introductory Programming course [3]. Python [27] is selected as its initially-featured programming language since it has been widely used as main programming language for various large computer science (CS) departments and online courses. Nevertheless, as PythonTutor is further developed, several popular programming languages such as Java and C++ are also incorporated to fulfill users' need. Unlike other program visualization tool, PythonTutor is designed as a web-based application so that users can access such tool anytime and anywhere as long as they are connected to the Internet. In addition, PythonTutor is also featured with responsive UI which can be viewed on the gadget. Users can open it from personal computers, laptops, tablets, or smartphones.

Even though there are many available PV tools and most of them have been proven to be effective, PV effectiveness is greatly relied on students' background, motive, and behavior. Thus, its impacts may be varied per university. As we know, each university has their unique student characteristics that may affect PV effectiveness. Based on the reason, this paper intends to evaluate the impacts of a PV tool when it is incorporated to Information Technology major, Faculty of Information Technology, Maranatha Christian University, Indonesia. Evaluation is conducted in two-fold which are questionnaire survey and quizzes. Both mechanism are incorporated after the students have been accustomed to use PV. For our case study, we select PythonTutor as our sample PV which is incorporated in our work.

III. Research Method

In general, there are three objectives that we want to address in this paper. Firstly, we want to measure the effectiveness of PV in assisting students for doing several sub-tasks in programming. Secondly, we want

Cite this article as: O. Karnalim and M. Ayub "The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor", CommIT (Communication & Information Technology) Journal 11(2), 67–76, 2017.

TABLE I
THE MEAN AND STANDARD DEVIATION, IN BRACKETS, OF THE RESPONDENT SCORES.

Attribute	Class A	Class B
Total Respondent	18	23
Admission test score	79.56 (10.52)	77.30 (4.61)
National test score for mathematics	55.00 (16.49)	55.87 (23.84)
Final written score	53.22 (24.04)	55.82 (23.61)
Final laboratory score	54.00 (20.86)	49.39 (19.11)

TABLE II
THE SCHEDULE OF PV USAGE FOR THE EVALUATED CLASSES.

Week	Class A	Class B
1st week: Void function		×
2nd week: Non-void function		×
3rd week: Array	×	
4th week: Advanced array	×	
5th week: Matrix		
6th week: Searching and Sorting	×	×

to measure the effectiveness of PV for advanced course topics such as array or function. Lastly, we want to summarize their feedbacks toward the usage of PV for teaching Introductory Programming. All objectives are measured through questionnaire survey and quiz result. Questionnaire survey is used to cover all mentioned objectives based on student's perspective, whereas quiz results are be used to cover the first two objectives based on academic grade.

To evaluate such aspects, students from two Introductory Programming classes in odd semester of 2016/2017 academic year are taken as our respondents. In fact, there are three classes of Introductory Programming course at that semester. However, since quiz-based evaluation incorporated in our work are measured by comparing two classes, only the classes are taken into our consideration. One class will act as an experimented class whereas the other one will be assigned as a control class. The statistics of involved classes can be seen in Table I. All attributes except the number of respondent are taken into our consideration based on our previous work about predicting student outcome on Introductory Programming [28]. These attributes have been used as learning features to detect the student outcome using data mining technique, namely J48. We believe that average and standard deviation of such attributes may be used as a prior knowledge to discuss our evaluation result further. Admission test score and national test score for mathematics are considered as pre-attributes, which are recorded before the students incorporated PV. Final written and laboratory score, on the other hand, are considered as post-attributes, which are taken after the students have incorporated PV. In general, class A outperformes class B in term of admission test score and final laboratory score. Yet, class A still yields lower score on the remaining attributes, national test score for mathematics, and final written score.

Since our evaluation requires a PV tool for our case study, PythonTutor [3], a web-based PV tool developed by Guo, is selected to be incorporated in our work. PythonTutor is preferred to other PV tools based on following reasons: 1) PythonTutor is designed as a web-based application with responsive UI. Thus, our students may use it anytime and anywhere as long as they are connected to the Internet; and 2) PythonTutors initial programming language is similar with our used programming language on Introductory Programming, which is Python.

To gain more accurate result, evaluation is only conducted after the respondents have used PV tool for six weeks alternately. The detail of PV usage for six weeks on two evaluated classes can be seen on Table II. On the first two weeks, class B acts as an experimented class which uses PV for learning function whereas class A acts as a control class. The assignment is swapped on the next two weeks. Class A acts as an experimented class which uses PV for learning array whereas the other one acts as a control class. The fifth and sixth week are used to measure the impact of PV for each class. The fifth week is used to represent initial state where both classes do not use PV and the sixth week is used to represent the state of both classes after their learning session have been intervened by PV. Both states then will be compared to generate delta value representing the impact of PV for each class.

Each time a session is intervened with the PV usage, the lecturer uses PV to explain the material. For instance, it can be providing an example about how the concept of array works through PV tool. In addition, after one-way session has been conducted in that occasion, the students will be asked to try the given PV through several code samples. They are required to answer several questions according to given code samples. In such manner, students may have their personal experience about using PV tool for learning Introductory Programming.

Empirical evaluation conducted in our work is split into twofold: questionnaire survey and quizzes. Questionnaire survey is used to measure the impact of PV based on students perspective whereas quizzes are used to measure such impact based on academic result.

Questionnaire survey is given at the end of the sixth week schedule for using PV in both classes. It consists of 14 questions which can be seen on Table III. For convenience, each question will be assigned with an unique ID and the ID will be used to refer that question

Cite this article as: O. Karnalim and M. Ayub "The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor", CommIT (Communication & Information Technology) Journal 11(2), 67–76, 2017.

TABLE III
THE QUESTIONS PRESENTED IN THE SURVEY.

Q	Statement
1	PV helps student to understand the execution flow of running program
2	PV helps student to understand how variable content changes
3	PV helps student to understand the sequence of method invocation
4	PV helps student to trace their own code
5	PV helps student to understand standard algorithms such as searching and sorting
6	PV helps student to learn Introductory Programming material in general
7	The interface incorporated in PythonTutor is descriptive and representative
8	The features incorporated in PythonTutor fulfill student necessity for learning Introductory Programming
9	PV is effective to help student learn array / function (depend on the class of given respondent)
10	PV is effective to help student learn searching
11	PV is effective to help student learn sorting
12	PV is effective to help student learn advanced modules in general
13	Please provide feedback(s) about PythonTutor's features
14	Please provide feedback(s) about PV usage on Introductory Programming class

at the rest of this paper. In general, these questions are split into three categories where each category corresponds to one of our major goal. The first category is focused on the impact of PV for assisting several sub-tasks in programming. Questions which fall into this category (Q1-Q8) should be answered in 7-point Likert scale where 1 represents strongly disagree, 2 as negatively disagree, 3 as positively disagree, 4 as neutral, 5 as negatively agree, 6 as positively agree, and 7 as strongly agree. The second category covers the effectiveness of PV in assisting advanced course materials. Questions in this category (Q9-Q12) should also be answered in 7-point Likert scale too. The third category, on the contrary, is represented as two open questions (Q13 and Q14) where students can answer it with a free text. This category is focused on collecting students feedback about the usage of PV on Introductory Programming class. Among proposed questionnaire questions, it is important to note that several questions use "PythonTutor" to refer the PV since these questions rely heavily on domain-specific characteristics such as interfaces and features. Such characteristics may vary based on incorporated PV.

Quiz result are incorporated in our evaluation to measure the impact of PV in term of enhancing students grade. In general, we will conduct four quizzes during six weeks of PV usage. The first two quizzes will be conducted at the end of the fourth week. They cover both function and array material where each quiz consists of four questions. On the contrary, the last two quizzes would be conducted at the end of the sixth week. Similar with the first two, each of them consists of four questions. Yet, the proportion of involved materials is quite different. They cover both matrix and searching and sorting instead of function and array.

Questions for each evaluated material is designed based on the first four level of Bloom taxonomy [29], which are remembering, understanding, applying, and analyzing. We do not incorporate all level of Bloom Taxonomy since the fifth and sixth level are not designed to undergraduate students. These levels are designed for the graduate ones. For each quiz, each question is weighted similar to each other in term of its grade proportion. In other words, for each quiz, a question will be weighted as 25 of 100 points.

IV. RESULTS AND DISCUSSION

A. Questionnaire Survey Result

The result of questionnaire survey is split into twofold which are the result of close and open questions. Close questions refer to questions which should be answered in 7-point Likert scale whereas open questions refer to questions which asked about student's feedback.

First of all, we will discuss about the result of close question category. It consists of 12 questions which cover our first two goals measuring the impact of PV for assisting students to do both sub-tasks in programming and specific advanced topics. The mean score of Q1-Q12 result toward our respondents can be seen in Figure 1. In general, all statements are roughly agreed by our respondents since, when it is viewed based on all respondents, all statements are scored higher than 5 and their mean score is 5.519. Among these statements, Q8 yields the highest Likert score, which is 5.853, and Q12 yields the lowest one, which is 5.097. In other words, they think that PythonTutors features are enough to learn Introductory Programming in general but some advanced modules are better to be learned through another tool such as Algorithm Visualization.

Class A students who get higher final laboratory score in average, Q8 still yields the highest score among all statements. Yet, the lowest-score statement is not Q12 anymore. There are three statements which shares the lowest score based on class A student feedbacks, Q2, Q3, and Q11. When it is discovered, Q2 yields a considerably low score since they think that showing all variable values on log may confuse students due to overwhelming information. The issue about overwhelming information also causes Q3 to yield the lowest score. They think that showing all method invocations, including the technical ones, may confuse the students since most students only care about their own-defined methods. Q11, on the contrary,

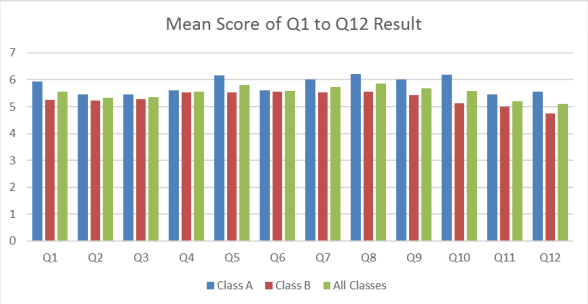

Fig. 1. The mean scores of Q1 to Q12.

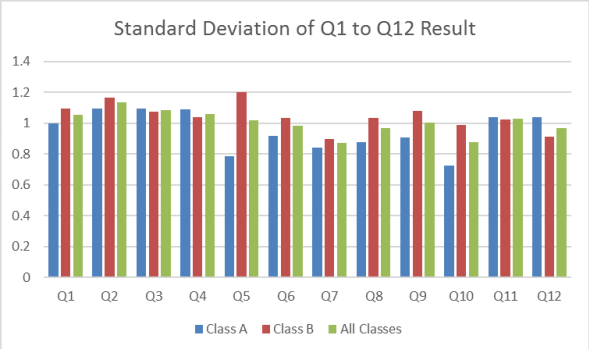

Fig. 2. The standard deviations of Q1 to Q12.

yields the lowest score since they prefer not to use PV for learning sorting. In PV, all information about execution trace are presented. Thus, students may not get ”the big picture” of such algorithm. They may get distracted easily with the overwhelming information. From class A students who get higher final laboratory score in average, Q8 still has the highest score among all statements. Yet, the lowest-score statement is not Q12. There are three statements which share the lowest score based on class A student feedbacks, Q2, Q3, and Q11. When we analyze this further, Q2 yields a considerably low score since they think that showing all variable values on log may confuse students due to overwhelming information. Such issue about overwhelming information also causes Q3 to yield the lowest score. They think that showing all methods invocations, including the technical ones, confuse the students since most students only care about their own-defined methods. Q11, on the contrary, has the lowest score since they prefer not to use PV for learning sorting. In PV, all information about execution trace are presented. Thus, students may not get ”the big picture” of such algorithm. They get distracted easily with the overwhelming information.

For class B students who get higher final written score in average, the phenomena, at some extent, are quite similar to class A findings, Q8 still yields the highest score whereas Q2, Q3, and Q11 yield a considerably low score. Yet, from the class B perspective, Q6 also shares the highest score. It generated 5.565 of 7, which is as high as Q8s mean score. They think that PV may be beneficial for learning Introductory Programming and in particular PythonTutor is a tool which features fulfilled such necessity. Despite the fact that Q2, Q3, and Q11 have a low score based on class B perspective, Q12 still yields the lowest one since they think that not all advanced modules are effective to be taught through PV.

In general, respondents from class A, who have higher final laboratory score, tend to provide higher score for all statements. This finding is natural since final laboratory score is obtained based on students capability about writing program implementation and PV is originally aimed to assist students for learning the task. Students who learn practically through programming implementation should feel that the tool is helpful since it help them to visualize the implementation in comprehensive manner. However, the impact of the tool is not too significant for students who learn programming in theoretical manner. This finding is deducted from the fact that respondents from class B, who have higher final written score, tend to provide lower scores for all statements compared to score given respondents in class A.

In order to measure the variation between respondents feedback, the standard deviation of each statement result is also presented on Figure 2. In general, the variance of the response is considerably small since most statements have standard deviation lower than 25% of its respective mean value. Among these statements, Q5 for class B respondents is the highest standard deviation which is 1.201 since there are two contradicting views about the impact of PV in learning standard algorithms. Some students say that PV is enough to teach standard algorithm whereas the others say that PV should be replaced with AV when it comes to learning standard algorithms. The lowest-scored statement, on the other hand, is Q10 for class A respondents. It is only 0.727 since all respondents in class A have similar perspective of the PVs impact toward searching algorithm. They positively agree that PV may help students for learning such topic even though searching is considered as a standard algorithm. This finding is logical since searching is the simplest topic in standard algorithms. Even though all information about the execution trace are provided on PV, students can still conveniently learn the material since the presented information does not overwhelm the students.

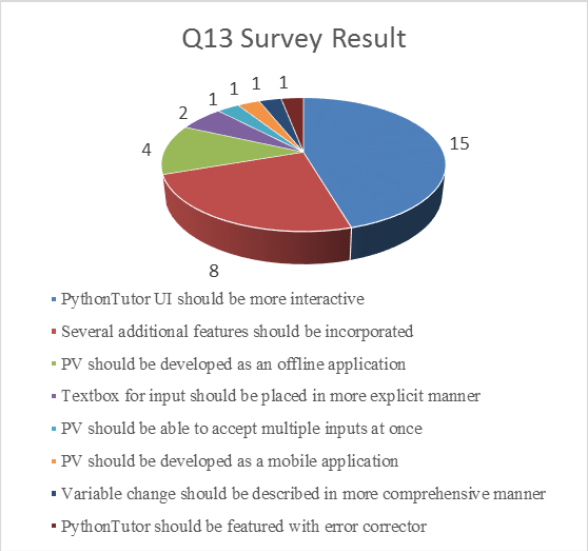

Fig. 3. The Q13 survey results.

Q13 and Q14 are incorporated to collect students feedback about given PV. Both of them are open questions where students can answer it freely. They can provide no answer, an answer, or more than one answer. However, to provide a concise result, our work summarizes and merges the similar feedbacks.

Q13 is used to collect students feedback about PythonTutors features. The detail of Q13 result can be seen in Figure 3. In general, there are 33 responses which are classified into eight categories. Among these categories, the top three categories take more than half of the responses. The highest category is about the interaction of PythonTutor. It explicitly states that the UI of PythonTutor should be more interactive to keep students focus. The respondents believe that providing colorful animation during program execution is an example to achieve such goal. The second highest category is about incorporating additional features to PythonTutor. The respondents state that natural language information and theoretical material of programming are necessary to assist students learning. The third highest category is about the offline version of PythonTutor. The respondents claim that the version is necessary due to limited internet access in Indonesia. They believe by providing the offline version may help students to learn without relying on Internet.

The remaining responses about PythonTutors features are only occurred once or twice. Firstly, it is about the input textbox of PythonTutor. The respondents agree that such textbox is quite hard to see since it is placed at the bottom of the code. They believe that it will be better to be interpreted as a pop-up alert. Secondly, it is about multiple input issue. PythonTutor accepts the input as the program is executed. Thus, it will be inconvenient when the program requires numerous inputs. The students should feed the input one by one via input textbox. The respondent believe that enabling user to provide all inputs at once through copy-and-paste technique should be beneficial. Students are only required to provide the inputs once at the beginning of program execution. Thirdly, it is about the mobile version of PythonTutor. Despite the fact that PythonTutor can be accessed via mobile web browser, the respondents state that converting the tool to mobile application is advantageous. Domain-specific mobile application may not only take less data usage but also provide higher capabilities compared to a web application opened through a browser. Web browser usually takes more bandwidth due to their generic connection mechanism and limits the capability of opened application. Fourthly, it is about variable change. The respondents state that the change should be delivered in more declarative manner such as highlighting the changed variable. Finally, it is about error correction. A respondent believes that PythonTutor should be featured with error correction so that students can learn how to fix an error while learning.

Q14 is used to collect students feedback about PV usage during Introductory Programming course. The detail of Q14 result can be seen in Figure 4. Different from Q13, Q14 yields less categories. It only generates three categories from 36 responses. Twenty six responses state that the frequency of PV usage during in-class session should be increased. The respondents argue that such tool may help students in term of learning programming used in each possible occasion during in-class session. Nine responses state that PV may also help the students during laboratory session. They argue that the tool may assist students to find logic error. One of the responses states that PV should not only be implemented on advanced introductory programming materials, but also on the basic ones such as input-output, branching, and looping. The respondents believe that PV is also beneficial for learning basic materials.

To sum up, the result of our questionnaire survey states that PV is a promising tool to assist student learning programming, particularly in Introductory Programming course. Despite the fact that several features on PythonTutor, our case study of PV tool is required to be developed further to fulfill students' need, the respondents still believe that the tool may be beneficial for learning programming. They even wish to increase the frequency of PV usage during the course.

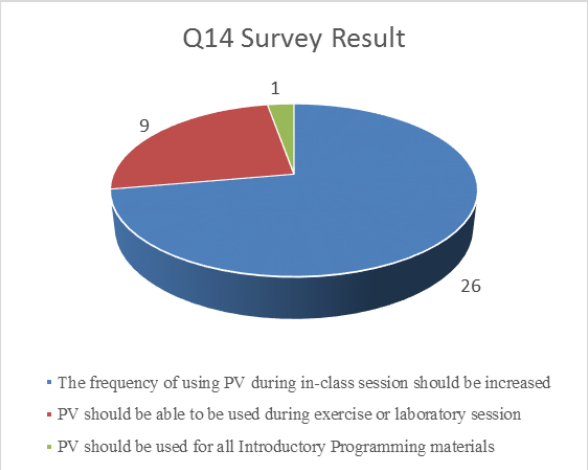

Fig. 4. The Q14 survey results.

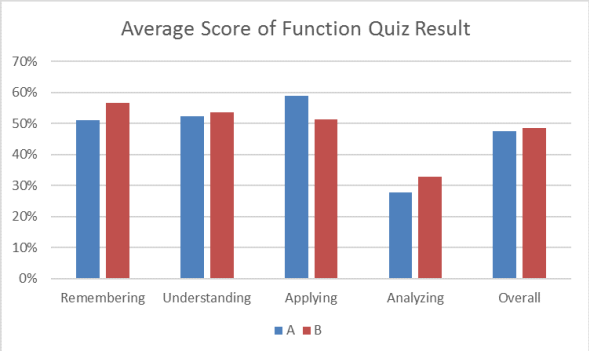

Fig. 5. The average scores of function quiz result.

B. Quiz Result

The quiz result discussed in threefold is aimed to measure the PV impact through students grade. The first two are focused on its impact for learning specific advanced module whereas the last one is focused on its impact for the students as its users. It is important to note that we assumed that every module has similar difficulty in this work. Therefore, no difficulty difference will be taken into our consideration in this evaluation.

To evaluate the PV impact for learning specific advanced modules, our work takes two advanced modules into account. These modules are function and array module. Based on our proposed schedule, class B becomes an experimented class and class A is a control class on function module. Then, the assignment is swapped for array module. Class A becomes an experimented class and class B is a control class.

The detail of student quiz result for function module can be seen in Figure 5 where the horizontal axis represents the questions and vertical axis represents the grade. As seen in Figure 5, class B gets higher score than A for most questions. Therefore, it can be concluded that PV has affected the learning process of function module positively since class B, which is assigned as the experimented class, has higher score than class A, which is as a control class. Among all questions, applying is the only one category where score of class B is lower than class A. The question given on that category is about function invocation where students are required to predict the output of a nested function invocation. From our perspective, we believe that class A respondents have got higher score on that question since most of them have known the concept of function call in math and nested function invocation is quite similar to the mechanism. The concept of function call is usually taught in math as a basic concept. In other words, when the students get an average score in math (about 50 of 100), it can be concluded that they have already known the concept. Based on national test score for math, respondents in class A have more balanced math skill than class B. Its standard deviation (16.49) is extremely lower than class B (23.84). Thus, it can be stated that more respondents in class A may have already known the concept of function call since their scores are less fluctuated on the middle score (50 of 100).

The detail of the quiz result for array module can be seen in Figure 6 where the horizontal axis represents the questions and vertical axis represents the grade. As seen in Figure 6, class B outperformes class A in the most questions despite the fact that class A is the experimented class in this module. The reduction is caused by PythonTutor input mechanism. In PythonTutor, users are asked to provide input one by one regarding to the execution flow. Even though the mechanism is beneficial for understanding how the program works, it is inconvenient when the program requires numerous inputs. The users are required to provide the input as they concentrate to understand the program flow. The issue affects the learning process of array module since most array-based programs require numerous inputs. These programs typically ask users to provide input for filling array elements manually. Thus, the focus of most students may be altered to provide input instead of understanding the program flow. They may not understand the array material better. We believe that the issue can be handled by providing multiple-input mechanism where students can store all required inputs right before the PV visualizes the program flow. This solution is also discussed in Q13 questionnaire survey result by two respondents.

To evaluate the PV impact for students (i.e. the first objective on this paper), our work takes two mod-

Cite this article as: O. Karnalim and M. Ayub “The Effectiveness of a Program Visualization Tool on Introductory Programming: A Case Study with PythonTutor”, CommIT (Communication & Information Technology) Journal 11(2), 67–76, 2017.

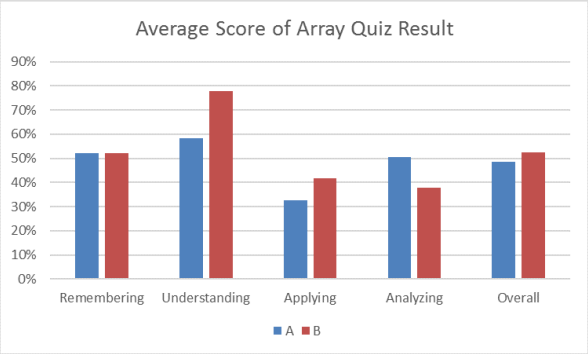

Fig. 6. The average scores of array quiz result.

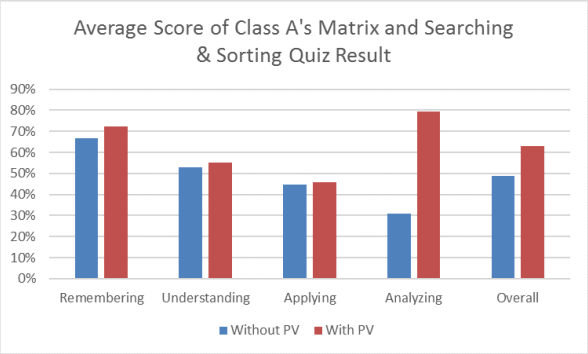

Fig. 7. The average score of Class A quiz result to measure the PV impacts.

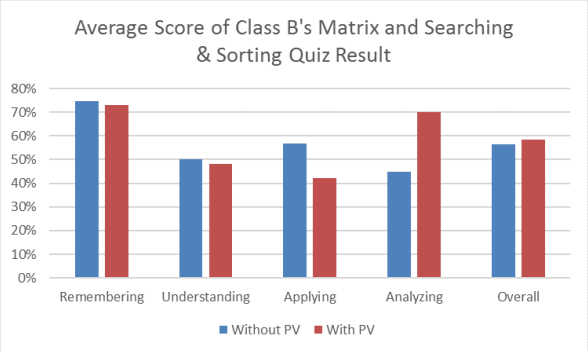

Fig. 8. The average scores of Class B quiz result to measure the PV impacts.

ules for comparison, matrix and searching and sorting module. In matrix module, the students are asked to learn without PythonTutor. In other words, they learn matrix material in conventional manner. In searching and sorting module, students are asked to learn with the assistance of PythonTutor. The PV impact for both classes is deducted by comparing the quiz result for matrix and searching and sorting module. Matrix quiz result is considered to represent student capabilities without the PV whereas searching and sorting quiz result is considered to represent student capabilities with the assistance of PV.

The detail of quiz result which represents the PV impact toward class A can be seen in Figure 7 whereas the result for class B is seen in Figure 8. The horizontal axis represents given questions while the vertical one represents the grade. ”without PV” series represents matrix quiz result whereas ”with PV” series represents searching and sorting quiz result. As seen in Figure 7, class A respondents get higher score when they incorporate PV. The highest improvement is in the Analysis question whereas the lowest one is in the applying question. However, such improvement is not explicitly occurred on class B. As seen in Figure 8, even though their ”with PV” score is higher in overall category, that score is low on the most specific categories. The issue is caused by the learning style of respondents in class B. According to final written score for both classes, the respondents are more accustomed to learn theoretically since they get higher grades in final written score than class A. Therefore, PV is originally aimed to assist students in practical manner, the tool is not beneficial for class B.

In conclusion, according to our evaluation, it is clear that learning function material can be supported by the assistance of PV. Function of quiz result is improved in most question categories. It even yields higher score in overall category. Nevertheless, such improvement is not explicitly occurred on array module since our case study PV, namely PythonTutor, is not featured with multiple-input mechanism. We believe that by providing such feature, students’ capabilities for learning array material would be improved through given PV. When perceived based on student capabilities, PV may help student to learn programming material. Yet, the improvement may become more significant when the students are accustomed to learn in practical manner, which is learning by doing.

V. Conclusion and Future Work

Based on our evaluation and questionnaire survey and quiz result, PV can help students to learn Introductory Programming course. We have measured its impacts through three objectives and all of them yield considerably good results. Firstly, we have measured the impacts of PV for assisting students to do several sub-tasks in programming such as understanding execution flow or variable change. The objective is measured through both survey and quiz result. Since most survey statements which correspond to this objective have positive feedbacks, and related quiz result generates a considerably improvement by the existence

of PV, it can be concluded that the first objective is achieved. Secondly, we have measured the impact of PV for learning advanced topics. It is also measured by questionnaire and quiz result. Despite the promising result of questionnaire, its improvement based on quiz result is satisfactory. It only yields prominent improvement on function module. Thirdly, we have summarized students feedbacks toward PV features and usage through survey. Despite the fact that several features are required to enhance student understanding further, most students argue that the tool may be beneficial for learning programming. They state that the PV should be used frequently so that students can be helped further.

Besides achieving the main objectives, our work also deducts several correlations between the PV impacts and student characteristics. Firstly, PV is more suitable for students who are accustomed to learn through practical fashion. Secondly, despite its low improvement, students who are accustomed to learn in theoretical manner are still helped by the existence of PV. Thirdly, students with high math skill can learn function easier since its basic concept has been learned in math.

For future works, we enhance the frequency of PV usage in a course and re-evaluate its impacts for learning programming. The impact of such PV is expected to become higher than our current result due to its higher PV usage frequency. In addition, we are also encouraged to develop a PV which fulfills all required features collected through our survey. Hopefully, PV may help students to learn programming more effectively and efficiently.

References

[1] C. Areias and A. Mendes, "A tool to help students to develop programming skills," in *Proceedings of the 2007 international conference on Computer systems and technologies*. ACM, 2007.

[2] 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.

[3] P. J. Guo, "Online python tutor: Embeddable web-based program visualization for cs education," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 2013.

[4] 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, 2017.

[5] O. Karnalim and E, "Interfacing complexitor: An empirical-based educational tool for learning time complexity," *Journal of IRD (Informatics Research and Development)*, vol. 1, no. 1, 2017.

[6] 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, 2011.

[7] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari, "Visualizing programs with jeliot 3," in *Proceedings of The Working Conference on Advanced Visual Interfaces*, 2004.

[8] P. Gestwicki and B. Jayaraman, "Interactive visualization of java programs," in *Symposia on Human Centric Computing Languages and Environments, 2002*, 2002.

[9] T. Rajala, M. J. Laakso, E. Kaila, and T. Salakoski, "Ville - a language independent program visualization tool," in *Proceedings of The 7th Baltic Sea Conference on Computing Education Research*, Finland, 2007.

[10] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. 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 SIGCSE Bulletin*, vol. 33, no. 4, 2001.

[11] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, B. Simon, and L. Thomas, "A multi-national study of reading and tracing skills in novice programmers," in *ACM SIGCSE Bulletin*, vol. 36, no. 4, 2004.

[12] J. D. Tenenberg, S. Fincher, K. Blaha, D. Bouvier, T. Y. Chen, D. Chinn, S. Cooper, A. Eckerdal, H. Johnson, R. McCartney, and A. Monge, "Students designing software: A multi-national, multi-institutional study," *Informatics in Education*, vol. 4, no. 1, 2005.

[13] S. M. Cisar, R. Pinter, D. Radosav, 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, 2011.

[14] R. E. Mayer, "Systematic thinking fostered by illustration in scientific text," *Journal of Educational Psychology*, vol. 81, no. 2, 1989.

[15] 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 (PVW08)*, vol. 9, no. 2, 2009.

[16] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards, "Algorithm visualization: The state of the field," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 3, 2010.

[17] S. Halim. (2015, May) Visualgo. [Online]. Available: http://visualgo.net/

[18] E. T. Y. Ling, "Teaching algorithms with web-based technologies," mathesis, B.Comp. Dissertation, Department of Computer Science, School of Computing, National University of Singapore, 2014.

[19] AlgoViz. (2015, Dec.) Algoviz.org : The algorithm visualization portal. [Online]. Available: http://algoviz.org/

[20] L. Christiawan and O. Karnalim, "P-asd1: An indonesian desktop-based educational tool for basic data structures," *Jurnal Teknik Informatika dan Sistem Informasi (JuTISI)*, vol. 2, no. 1, 2016.

[21] 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 (SNATI)*, Yogyakarta, 2016.

[22] 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, 2017.

[23] E. Kaila, T. Rajala, M. J. Laakso, and T. Salakoski, "Effects of course-long use of a program visualization tool," in *Proceedings of the Twelfth Australasian Computing Education Conference*, vol. 103. Brisbane: Australian Computer Society, Inc., 2010, pp. 97–106.

[24] O. Kannusmäki, A. Moreno, N. Myller, and E. Sutinen, "What a novice wants: Students using program visualization in distance programming course," in *Proceedings of the Third Program Visualization Workshop (PVW'04)*, Warwick, 2004, pp. 126–133.

[25] S. M. Čisar, R. Pinter, D. Radosav, and P. Čisar, "Software visualization: The educational tool to enhance student learning," in *MIPRO, 2010 Proceedings of the 33rd International Convention*. IEEE, 2010, pp. 990–994.

[26] T. Rajala, M. J. Laakso, E. Kaila, and T. Salakoski, "Effectiveness of program visualization: A case study with the ville tool," *Journal of Information Technology Education : Innovation in Practice*, vol. 7, 2008.

[27] P. S. Foundation. (2016, Dec.) Welcome to python.org. [Online]. Available: https://www.python.org/

[28] 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, 2017.

[29] B. Bloom and D. Krathwohl, *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*. Addison Wesley, 1956.