Research Article

PV2P: Program Visualization and Pair Programming to Improve Students Understanding of Python Programming

Syahrul Affendi Abdul Rahman[1, *]

[1] Maktab Rendah Sains MARA Kuala Klawang; syahrul.affendi@mara.gov.my; ORCID ID (https://orcid.org/0009-0006-2802-7805/)

* Correspondence: syahrul.affendi@mara.gov.my; +606-6112561.

Abstract: *The objective of this research was to enhance the understanding of Python programming among students. The research involved 71 students from two form 2 classes at MRSM Tun Ghazali Shafie who were studying Basics of Computer Science (ASK). The data collection is conducted through observation, pre- and post-tests, and questionnaires. Previously, students encountered difficulties in creating accurate computer program flowcharts and generating program code from the given flowchart. The students also could not solve the complex computer programming problems on their own. The pre-test analysis revealed a mean score of 67.9 percent. Researchers evaluated the students' learning and introduced Program Visualization and Pair Programming (PV2P) method to enhance their understanding of Python programming. As a result, the post-test scores showed an increase to 85.3 percent, and the students responded positively to the questionnaires. Thus, PV2P was successful in improving the students' understanding of Python programming.*

Keywords: Program Visualization; Pair Programming; Algorithms; Collaborative; Python Programming.

DOI: 10.5281/zenodo.7800407

1. INTRODUCTION

Over the years, studies have shown that programming has proven to be a challenging task for many people (Stephen et al., 2011). Programming is a difficult subject to learn, even for novice students (Bennedsen & Caspersen, 2007). This causes students to be less motivated to learn it. The subject requires high problem-solving skills. In addition, it requires students to have the ability to think visually to interpret the abstract structure of the program (Ahmad Rizal et al., 2011). Furthermore, students often make mistakes in completing their programming, especially if they do it alone (Laurie & Richard, 2001). Program visualization (PV) is one of the various methods developed over the years to aid novices with their difficulties in learning to program. PV tool does improve students' experience of learning programming by helping them to tackle the abstractness of programming structure (Stephen et al., 2011) and program execution (Rajala et al., 2008). As a result, program visualization helps students to design programs, in addition to helping them to strategize solutions to problem solving in programming (Agno-balabat & Rojo, 2012). However, the usage of learning aid tools alone, such as PV could not have an effective impact on student programming performance. Hence, an active learning strategy should be used, in complement to PV tools to make the learning experience more encouraging (Derus, 2014; Mikko-Jussi et. Al., 2009). Hence, there have been several attempts to integrate visualization techniques and active learning into programming courses.

Research has found that integration of active learning and visual/verbal techniques may enforce may enhance cognitive knowledge and promote logical thinking skills with respect to programming performance (Hui, 2011). As there is little initiative on implementing PV tools and pair programming in teaching and learning programming subjects, especially in secondary education, this research intents to fill that gap. Hence, this research will investigate the effect of PV tools and pair programming on student performance and motivation of learning programming in ASK subject.

2. LITERATURE REVIEW

Program visualization (PV) can be defined as visual representation of program or algorithm execution in the form of graphical components (Rosminah, 2013). PV main goal is to visualize program execution which includes variable values, line-by-line statements implementation and program status. There are two main categories of PV which include dynamic PV or static PV. Dynamic PV can visualize program flow control by highlighting programming runtime execution. On the other hand, static PV focuses on visualizing code structure using pictures or diagrams such as flowchart (Rajala et al., 2008). Examples of dynamic program visualization tools are Jeliot3 & ViLLE ((Rajala et al., 2008). Examples of static program visualization tools are BlueJ (Kölling, 2003) and Flowgorithm (Cook, 2015). PV is more beneficial in learning programming because they display information in a manner that is familiar to users' mental representations of matters and allow data to be handled in a format that is closer to how things are managed in the real world; they are also simpler to comprehend for programming beginners (Basigie, 2022). PV able to solve some of the problems in learning computer programming, such as difficulty developing program algorithms, transferring algorithms to programming language and understanding program structures (Kadar et al., 2021). Researchers and practitioners have started to integrate collaborative components into programming activities because of their benefits, especially for students (Zakaria et el., 2021). The example of a collaboration activity that can be integrated into programming class is pair programming (Echeverria, 2019). Pair programming is one of the Extreme Programming (XP) agile software methodologies used in the industry with the main goal of improving interaction between two programmers to improve software quality (Vinod, 2014). Pair programming involves two programmers working collaboratively on one computer, one as a driver who operates the keyboard, concentrates on the lower-level details of the task, and another as a navigator who observes the driver. (Radhakrishnan, 2017). Through the process, the pairs swap responsibilities so that both partners become involved in the brainstorming process (Albayrak, 2022). Pair Programming enables the students to work together to solve complex programming problems, improve computational thinking, and develop real-world problem-solving skills (Weiqi et al, 2023). Pair programming indeed gives benefits to students' attitude, learning and academic performance (Faja, 2014).

3. METHOD & MATERIAL

3.1 Material and Development

The development of learning aid (BBM) for unplugged PV2P using several materials such as magnetized white board, laminate plastic, magnet strip and paper. On the other hand, Flowgorithm software was used for plugged PV2P where 2 students share a computer installed with the mentioned software for pair programming.

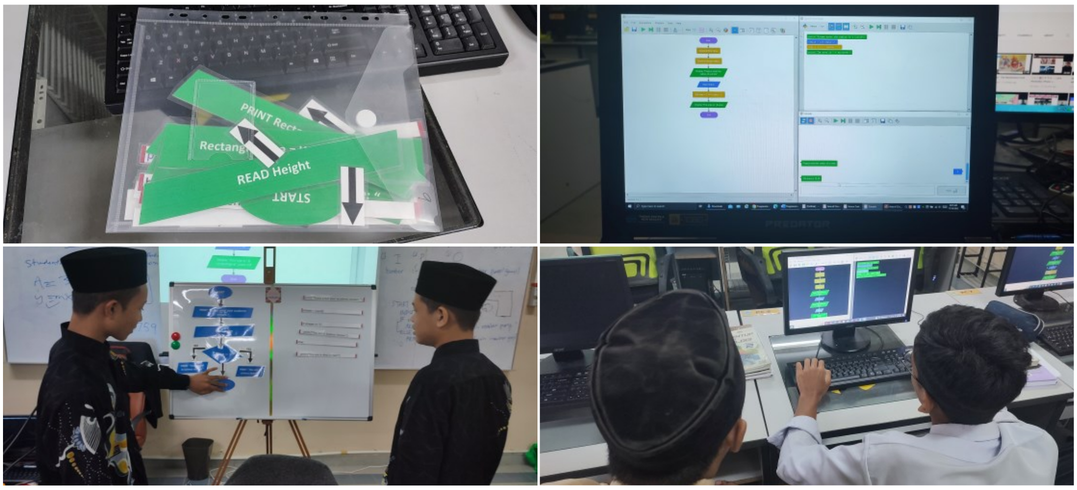

Figure 1. Unplugged PV2P (left) and plugged PV2P (right).

3.2 Methodology

This study involved a total of 71 students from two form 2 classes at MRSM Tun Ghazali Shafie who took the subject of Basic of Computer Science. Firstly, the students sat for a pre-test where they needed to answer about 6 questions about flow charts and Python programming. After that, the students' scores were recorded for later analysis. Then the students underwent unplugged and plugged about a month each in sequence. After that, the students sat for post-test for comparison with the earlier test. Finally, the students were given questionnaires to measure their responses to the new teaching method of PV2P.

4. FINDINGS

The results of the data collected through pre-test, post-test, and questionnaires.

4.1 Pre and Post-Test

Table 1. Pre-test and post-test result analysis.

Paired Sample Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Pre-Test	67.6901	71	6.98466	.82895
	Post-Test	85.2958	71	4.38307	.52017

Paired Sample Statistics

		N	Correlation	Sig.
Pair	Pre-Test & Post-Test	71	.191	.110

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	95% Confidence Interval of the Difference Upper			
Pair 1	Pre-Test Post-Test	-17.60563	6.98466	.82895	-19.38163	-15.82963	-19.771	70	.000

The value of t is 19.771 and the value of p is < .00001. The result is significant at p < .05.

4.2 Questionnaires - Closed Ended

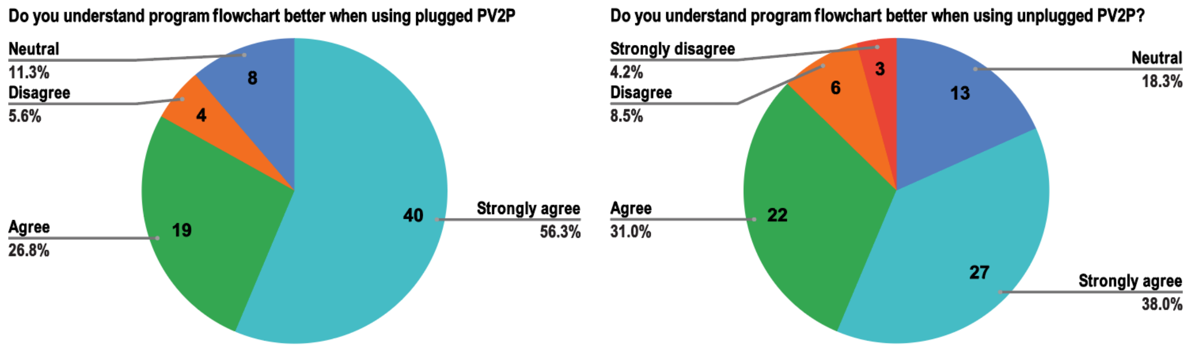

Figure 2. Pie chart of students' understanding about flowchart by using PV2P.

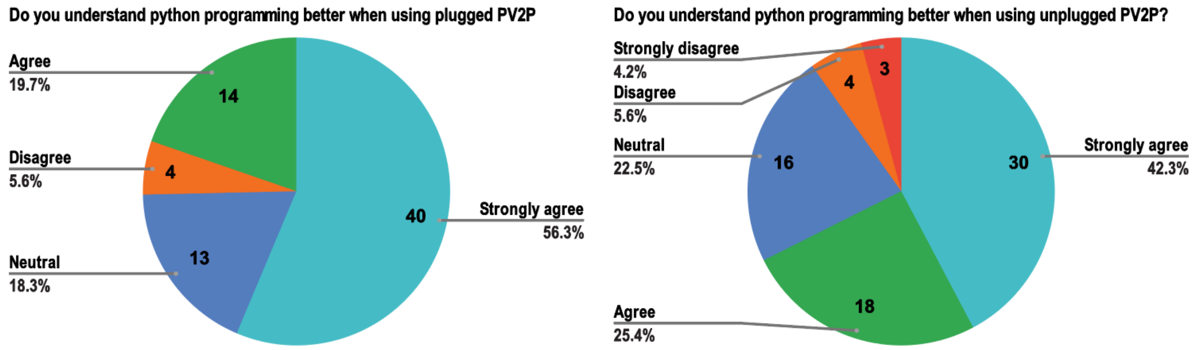

Figure 3. Pie chart of students' understanding about Python Programming by using PV2P.

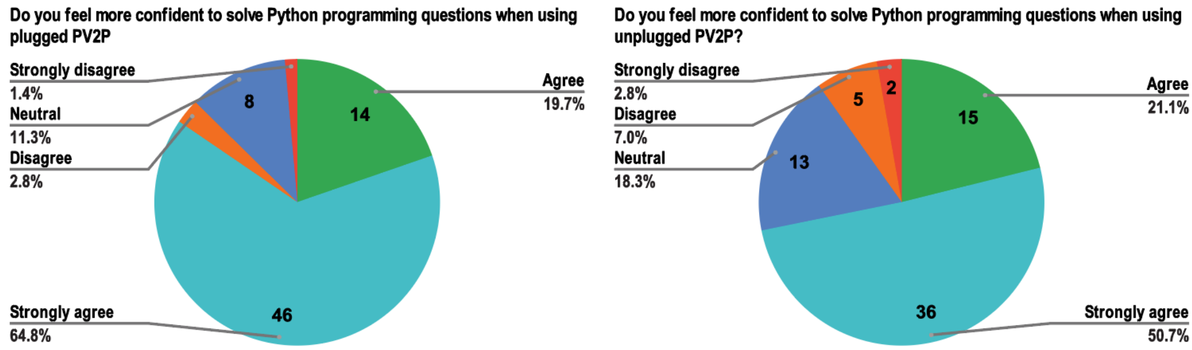

Figure 4. Pie chart of students' confidence about solving Python Programming questions by using PV2P.

4.3 Questionnaires - Open Ended

The summary of students' opinions about PV2P is shown below:

i. It's easier to understand flowcharts & Python programming by using PV2P.
ii. It is easier to understand the concept of Python programming (unplugged) before switching to using a computer (plugged).

iii. The use of PV2P in the classroom is fun.
iv. The use of PV2P in the classroom should continue.
v. PV2P helps reduces answer errors.
vi. PV2P promotes trust and cooperation in groups.

5. DISCUSSION

This study introduces a Teaching and Facilitation Session (PdPc) method by combining program visualization (PV) and pair programming known as PV2P. The results show that PV2P allows the students to obtain higher scores on tests due to better understanding and confidence in learning Python programming. The PV helps the student to grasp the essential concepts of algorithms before moving on to advanced concepts of programming such as syntax and program structure. This helps flatten the learning curve for the students to become familiar with Python programming. The cooperation introduced in pair programming enables the students to be more confident because they get help from their peers to solve problems together. This can promote the students to be 21st learners, which includes communicators, and collaborators.

6. CONCLUSION

In conclusion, PV2P has improved students' understanding of Python programming. Unplugged PV2P was developed by using a magnetized whiteboard where the student can arrange flowchart shapes and Python programming snippets cut out on top of it. Plugged PV2P was developed with the use of Flowgorithm software installed on a computer shared by a pair of students. The effectiveness of PV2P was measured from pre-test, post-test, and questionnaires. It has been proven that PV2P was able to increase student performance during the tests. In addition, the students became more confident in tackling questions about algorithms and Python programming. As a result of the experience of implementing this action research, it can be concluded that PdPc has been able to help students, especially in the subject of ASK. This creative and innovative PdPc approach can bring a big change in student achievement. Therefore, teachers need to diversify PdPc techniques and software to produce a positive impact on students. Students' acceptance of knowledge and skills is varied, not to mention involving technology. Improvements in PdPc must always be made to improve the quality of teaching and learning. Therefore, teachers need to explore new knowledge, skills, and technology constantly to dignify the professionalism of educators.

Acknowledgments: The author would like to thank to all students and teachers from MRSM Tun Ghazali Shafie and MRSM Kuala Klawang for contributed and helped to complete this research.

References

Agno-balabat, A. C. G., & Rojo, J. N. N. (2012). A Program Visualization Approach in Developing an Interactive Simulation of Java Programs for Novice Programmers, 10, 63–80.

Ahmad Rizal, M., Mohd Yusop, A. H., Abdul Rasid, R., & Mohamad Zaid, M. (2011). The Effect of Using Learning Model Based on Problem Solving Method on Students with Different Cognitive Style and Logic Ability. Educational Research, 2(9), 1498–1505.

Albayrak, Ebru & Polat, Elif. (2022). Pair Programming Experiences of Prospective Information Technologies Teachers. Bartın University Journal of Faculty of Education. 342-354. 10.14686/buefad.991448.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in Introductory Programming. ACM SIGCSE Bulletin, 39(2), 32–36.

Cook, D. D. (2015). Flowgorithm: Principles for teaching introductory programming using flowcharts. In Proc. American Society of Engineering Education Pacific Southwest Conference (ASEE/PSW) (pp. 158-167).

Derus, S. R. (2014). Integration Of Visualization Techniques and Active Learning Strategy in Learning Computer Programming: A Proposed Framework, (January), 93–103.

Echeverria, Leovy & Cobos, Ruth & Morales, Mario. (2019). Improving the Students Computational Thinking Skills with Collaborative Learning Techniques. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje. PP. 1-1. 10.1109/RITA.2019.2952299.

Faja, Silvana. (2014). Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses. Information Systems Education Journal. 12. 36-44.

Hui, T. H. (2011). Pair Programming and LSs in Computing Education: Its Impact on Students' Performances, 5, 613–626.

Kadar, R., Wahab, N. A., Othman, J., Shamsuddin, M., & Mahlan, S. B. (2021). A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review. International Journal of Academic Research in Progressive Education and Development, 10(3), 591–605.

Kölling, Michael & Quig, Bruce & Patterson, Andrew & Rosenberg, John. (2003). The BlueJ System and its Pedagogy. Computer Science Education. 13. 10.1076/csed.13.4.249.17496.

Laurie Williams and Richard L. Upchurch. (2001). In support of student pair-programming. SIGCSE Bull. 33, 1 (March 2001), 327–331

Mikko-Jussi Laakso, Niko Myller, & Ari Korhonen. (2009). Comparing Learning Performance of Students Using Algorithm Visualizations Collaboratively on Different Engagement Levels. Journal of Educational Technology & Society, 12(2), 267–282.

Mtakho Adam Basigie (2022). The Impact of Combining Follow-Up Questions and Worked Examples in Program Visualization Tool On Improving Students' Held Mental Models Of Pointers' Value And Address Assignment. Education & Pedagogy Journal, (2 (4)), 53-64.

Padmanabhuni, V. V. K., Tadiparthi, H. P., & Muralidhar Yanamadala, S. M. (2012). Effective pair programming practice-an experimental study. Journal of Emerging Trends in Computing and Information Sciences, 3(4), 471-479.

Radhakrishnan, P. & Kanmani, Selvadurai & Malaiyappan, Nandhini. (2017). Pair programming: A teaching and learning tool for defending student's mental energy. 14. 152-158.

Rajala, T., Laakso, M.-J., Kaila, E., & Salakoski, T. (2008). Effectiveness of Program Visualization: A Case Study with the ViLLE Tool. Journal of Information Technology Education, 7, 15–32.

Rosminah, S. (2013). Integration Of Visualization Techniques and Active Learning Strategy in Learning Computer Programming: A Proposed Framework. IJONTE, (5)1.

Stephen, M., Franklin, W., Elizabeth, A., Juma, K., & Patrick, O. (2011). Teaching Computer Programming in the 21 st Century, 1(6).

Xu, Weiqi & Wu, Yajuan & Ouyang, Fan. (2023). Multimodal learning analytics of collaborative patterns during pair programming in higher education. International Journal of Educational Technology in Higher Education. 20. 8. 10.1186/s41239-022-00377-z.

Zakaria, Zarifa & Vandenberg, Jessica & Tsan, Jennifer & Boulden, Danielle & Lynch, Collin & Boyer, Kristy & Wiebe, Eric. (2021). Two-Computer Pair Programming: Exploring a Feedback Intervention to improve Collaborative Talk in Elementary Students. Computer Science Education. 32. 1-28. 10.1080/08993408.2021.1877987.