Novice Programming Misconceptions in Elementary Education

Monika Mladenović[1][0000-0002-5330-605X] and Žana Žanko[1][0000-0002-9566-5093]

[1] University of Split Faculty of Science Department of Computer Science, Split, Croatia

{monika.mladenovic, zana.zanko}@pmfst.hr

Abstract. Programming misconceptions have been studied since the 1980s, when the first significant scientific papers on the topic emerged. Although much has changed since then—particularly in the technology surrounding us—novice programming misconceptions have remained largely consistent, regardless of age, programming language, or time period. However, most existing studies have been conducted at the university level, and there is still a lack of research focused on younger learners. As programming becomes an increasingly integral part of Informatics and Computer Science curricula in elementary schools, it is important to investigate misconceptions among young learners. Over the past decade, we have conducted four studies with fifth- and sixth-grade students, focusing on misconceptions related to the text-based programming language Python and the procedural programming paradigm. These studies were carried out in real classroom settings, involving a total of 435 students across four school years, 25 classes, five schools, and five teachers. In this paper, we present 17 identified programming misconceptions related to basic programming constructs: variables, sequencing, conditionals, and loops.

Keywords: Programming, Misconceptions, Programming Novices, Python, Elementary education, K-12.

1 Introduction

When interest in learning programming began to grow in the 1980s, scientific research in computer science education also emerged. Some of the earliest studies on programming misconceptions were conducted at the university level using languages such as PASCAL [1] and BASIC [2], [3], [4]. Today, BASIC has been largely replaced by Python in introductory programming courses. Although Python supports multiple paradigms, teaching at the elementary level typically follows the procedural paradigm, introducing students to core algorithmic structures such as sequencing, conditionals, and loops. These are supported by foundational programming constructs including variables, `if-else` statements, and `for` or `while` loops.

Importantly, learning to program is not merely a matter of learning syntax or mastering a programming language. Many misconceptions identified in earlier studies—despite differences in age group, context, and language—continue to appear in modern

programming classrooms. This suggests that difficulties in learning to program may be more cognitive than technical in nature.

In this paper, we present findings from four quasi-experimental longitudinal studies conducted over a period of four school years. Our aim was to identify and analyze frequent programming misconceptions among fifth- and sixth-grade (10-11 years old) students using Python. These misconceptions span basic programming concepts such as variables, sequencing, conditionals, and loops. We compare our findings with earlier research and highlight the persistent nature of many misconceptions, despite significant changes in curriculum, tools, and learner demographics.

2 Analysis of the results

Over four school years, we designed and conducted four quasi-experimental longitudinal studies, each targeting programming misconceptions among novices in elementary schools. Each study was approached from a different perspective and with different limitations, contributing to the triangulation of the research. All studies were conducted in authentic classroom settings, involving a total of 435 students across four school years, 25 classes, five schools, and five teachers. These studies were conducted as part of the second author's doctoral research, under the mentorship of the first author. Misconceptions were identified by analyzing students' test results following each study. To differentiate random errors from actual misconceptions, we applied a threshold: misconceptions were considered *frequent* if they appeared in 10–20% of responses, and *fairly frequent* if they appeared in more than 20%.

The first study was conducted during the 2015/2016 school year, involving eight classes and a total of 127 fifth- and sixth-grade students. We designed a test consisting of eight tasks based on previously identified misconceptions related to variables. The study compared students' understanding of variables and sequencing using the Python and Logo programming languages. Four misconceptions (M1, M2, M3, and M4) related to variables were identified [5], [6].

The second study was carried out in the 2016/2017 school year with 98 fifth-grade students across six classes. This study compared the results of an experimental group and a control group. In the experimental group, visualization techniques were used to introduce the basic programming constructs of sequencing and variables. Based on the results of the first study, the test was expanded to include 13 tasks related to variables. In addition to the four previously identified misconceptions, three new misconceptions (M5, M6, and M7) were found. We also observed that while misconceptions M2 and M4 remained consistent in frequency, the occurrence of M1 and M3 was halved through minor instructional interventions, where teachers paid special attention to addressing these misconceptions during lectures [7].

The next two studies focused on the transition from block-based to text-based programming languages. For these studies, we developed a new instrument that extended the previous one by including tasks related to conditionals and loops. The instrument consisted of 20 multiple-choice questions (MCQs) and 9 open-ended tasks. Among the

MCQs, 8 targeted variables, 6 addressed conditionals (*if-elif-else* statements), and 6 focused on the use of `for` loops.

The third study was conducted during the 2017/2018 school year with 47 sixth-grade students across 3 classes. The results confirmed the presence of five variable-related misconceptions (M1–M5) identified in the first two studies, along with two misconceptions related to conditionals (M8 and M9), and five related to loops (M10–M14) [8].

The fourth study was carried out in the 2019/2020 school year with 163 sixth-grade students across 8 classes. In addition to confirming the previously identified misconceptions (M1–M14), three new misconceptions were detected: one related to sequencing (M15) and two related to conditionals (M16 and M17). Several tasks involving loops also revealed variable-related misconceptions, such as M1*, M3*, and M13* [9].

Appendix A presents an overview of all detected misconceptions.

3 Conclusion

This paper summarizes four classroom-based studies exploring programming misconceptions among novice elementary school students learning text-based programming in Python. Across all studies, a total of 17 distinct misconceptions were identified, primarily related to variables, sequencing, conditionals, and loops. The findings highlight the importance of addressing specific misconceptions early in programming education and designing assessments and teaching strategies that explicitly confront them. Future work should focus on developing and evaluating teaching methods that can effectively mitigate these misconceptions in various learning environments.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] E. Soloway and K. Ehrlich, ‘Empirical Studies of Programming Knowledge’, *IIEEE Trans. Software Eng.*, vol. SE-10, no. 5, pp. 595–609, Sep. 1984, doi: 10.1109/TSE.1984.5010283.

[2] P. Bayman and R. E. Mayer, ‘A diagnosis of beginning programmers’ misconceptions of BASIC programming statements’, *Communications of the ACM*, vol. 26, no. 9, pp. 677–679, Sep. 1983, doi: http://doi.acm.org/10.1145/358172.358408.

[3] R. T. Putnam, D. Sleeman, J. A. Baxter, and L. K. Kuspa, ‘A Summary of Misconceptions of High School Basic Programmers’, *Journal of Educational Computing Research*, vol. 2, no. 4, pp. 459–472, Nov. 1986, doi: 10.2190/FGN9-DJ2F-86V8-3FAU.

[4] B. Du Boulay, ‘Some Difficulties of Learning to Program’, *Journal of Educational Computing Research*, vol. 2, no. 1, pp. 57–73, Feb. 1986, doi: 10.2190/3LFX-9RRF-67T8-UVK9.

[5] M. Mladenović, Ž. Žanko, and I. Boljat, 'Programming Misconceptions at the K-12 Level', in *Encyclopedia of Education and Information Technologies*, A. Tatnall, Ed., Cham: Springer International Publishing, 2019, pp. 1–13. doi: 10.1007/978-3-319-60013-0_234-1.
[6] Ž. Žanko, M. Mladenović, and I. Boljat, 'Misconceptions about variables at the K-12 level', *Education and Information Technologies*, vol. 24, no. 2, pp. 1251–1268, Oct. 2019, doi: 10.1007/s10639-018-9824-1.
[7] Ž. Žanko, M. Mladenović, and D. Krpan, 'Analysis of school students' misconceptions about basic programming concepts', *Computer Assisted Learning*, vol. 38, no. 3, pp. 719–730, Jun. 2022, doi: 10.1111/jcal.12643.
[8] Ž. Žanko, M. Mladenović, and D. Krpan, 'Mediated transfer: impact on programming misconceptions', *J. Comput. Educ.*, vol. 10, no. 1, pp. 1–26, Mar. 2023, doi: 10.1007/s40692-022-00225-z.
[9] M. Mladenović, Ž. Žanko, and G. Zaharija, 'From Blocks to Text: Bridging Programming Misconceptions', *Journal of Educational Computing Research*, vol. 0, no. 0, p. 07356331241240047, 2024, doi: 10.1177/07356331241240047.

A Appendix

Misconception	**Programming Concept**	**Example in Python**	**Explanation**
M1: Assigning expression variables instead of a calculated value	Variables	`a = a + 1`	Students believe that the variable contains the unevaluated expression as a string (*a + 1)*.
M1*: Same as M1 in the loop	Variables, Loop	`for i in range(0, 6):` `  print(i + 1)`	Students believe the expression (*i + 1*) will be printed as a string six times.
M2: Believing a variable stores the sum of all its previously assigned values	Variables	`a = 100` `a = 20`	Students think the variable stores the sum of all its assigned values (i.e., 120).
M3: Using the symbolic name of a variable instead of its value	Variables	`a = 100` `print(a)`	Students expect the name of the variable *a* to be printed, not its value.

M3*:
Same as M3 in the loop
	Variables, Loop	`for i in range(0, 4):`
		`  add = i`
		`print(add)`
			Students believe the variable name (*add*) will be printed instead of its value (3).

M4: Using the first (or previous) value assigned to a variable	Variables	`a = 100` `a = 20`	Students believe the variable *a* still holds the first value (100).
M5: Data type confusion	Variables	`a = 100` `print('a')`	Students believe the variable *a* contains the numeric value 100, although the letter '*a*' is printed.
M5*: Same as M5 in the loop	Variables, Loop	`for i in range(0, 3):` `  print('i+1')`	Students believe that (*i+1*) will be evaluated and printed.

M6:
Expecting that variables will be printed in assignment values order
	Variables	`y = 200`
		`x = 1`
		`print(x, y)`
			Students expect values to be printed in the order of assignment (200 1), rather than in the order specified in the print statement (1, 200).

M7:
Incorrect variable swap
	Variables	`a = 20`
		`b = 100`
		`a = b`
		`b = a`
		`print(a, b)`
			Students believe the values of *a* and *b* have been swapped (100 20).

M8:
Expecting the input value to be printed in a single-selection if statement
	Conditionals	`t = int(input())`
		`if t >= 20:`
		`  print('Not cold!')`
			Students believe that the input value of the variable *t* will be printed, regardless of whether the condition is true or false, in a single-selection *if* statement (no *else* branch).

M9:
Incorrect interpretation of a boundary condition
	Conditionals	`s = int(input())`
		`if s < 128:`
		`  print('I cannot see!')`
		`else:`
		`  print('I see!')`
			Students believe the condition is true when the boundary value (128) is entered.

M10: Including the final value in the loop range	Loop	`for i in range(0, 6):`	Students believe the loop ends with the value 6.
M11: Loop starts at 1	Loop	`for i in range(0, 6):`	Students believe the loop starts at 1.
M12: Ignoring the repetition in a *for* loop	Loop	`for i in range(0, 6):` `  print('+')`	Students believe the string will be printed only once.

M13:
Variable name affects its value
	Variables, Loop	`add = 0`
		`for i in range(0, 6):`
		`  add = i`
			Students believe the variable *add* stores the final value of the loop range (6).

M14: Ignoring the summing in the loop	Loop	`for i in range(0, 6):` `  add = add + i`	Students believe the variable *add* contains the loop range value (6), ignoring the accumulation of the values within the loop.

M15:
Reversing execution order of statements
	Sequence	`b = 0`
		`a = b + 100`
		`b = 100`
		`print(a)`
			Students believe the final value of *b* (100) is used when calculating *a*, leading them to expect the result to be 200.

M16:
Misinterpreting the condition value
	Conditionals	`t = 30`
		`if t >= 20:`
		`  print('Not cold!')`
			Students believe the condition is false even when it is true, leading them to expect no output.

M17:
Ignoring the else branch
	Conditionals	`s = int(input())`
		`if s < 128:`
		`  print('I cannot see')`
		`else:`
		`  print('I see!')`
			Students believe that if the condition is false, nothing will be printed.