ISSEP 2025

Novice Programming Misconceptions in Elementary Education

Monika Mladenović and Žana Žanko
University of Split, Faculty of Science, Department of Computer Science
{monika.mladenovic, zana.zanko}@pmfst.hr

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.

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.

ACTIVITY OVERVIEW

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 different schools, and five different teachers.

Phase 1

*Comparison of students' final test results after learning with **Logo** vs. **Python*** [5], [6] .

2015./16.

Participants: 127
Classes: 8
Grades: 5–6
Schools: 3
Teachers: 2

Phase 2

*Comparison of student achievement using **visualization** techniques with **Python*** [7].

2016./17.

Participants: 98
Classes: 6
Grade: 5
Schools: 3
Teachers: 2

Phase 3

*Impact of using **MakeCode** (block-based) for **mediated transfer** of programming concepts to **Python*** [8].

2017./18.

Participants: 47
Classes: 3
Grade: 6
Schools: 1
Teacher: 1

Phase 4

*Comparison of **MakeCode** (block-based) for **mediated transfer** vs. traditional **Python** teaching of programming concepts* [9].

2019./20.

Participants: 163
Classes: 8
Grade: 6
Schools: 3
Teachers: 2

DETECTED MISCONCEPTIONS

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.

ID	Misconception	Concept(s)	Python Example	Explanation	Phase 1	Phase 2	Phase 3		Phase 4	
							Post-Python	Post-MakeCode	Control	Experim.
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*).	46,60%	21,43%	22,34%	6,38%	21,67%	1,37%
M1*	Same as M1 in 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.	—	—	40,43%	23,40%	20,00%	4,11%
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).	21,20%	21,09%	16,31%	3,55%	14,17%	2,06%
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.	17,78%	8,16%	25,53%	14,89%	10,56%	0,68%

M3*	Same as M3 in 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).	—	—	—	—	10,56%	0,00%

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).	16,98%	17,69%	12,77%	4,26%	16,39%	8,56%
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.	—	36,74%	14,89%	2,13%	—	—
M5*	Same as M5 in loop	Variables, Loop	`for i in range(0, 3):` `  print('i+1')`	Students believe that (*i+1*) will be evaluated and printed.	—	—	—	—	11,11%	1,37%

M6	Expecting that variables will be printed in assignment values order	Sequence	`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).	—	37,76%	—	—	—	—

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).	—	13,78%	—	—	—	—

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).	—	—	21,28%	10,64%	12,41%	0,23%

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.	—	—	23,40%	23,40%	16,11%	14,38%

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.	—	—	42,55%	24,11%	26,67%	15,75%
M11	Loop starts at 1	Loop	`for i in range(0, 6):`	Students believe the loop starts at 1.	—	—	23,40%	8,51%	13,33%	2,74%
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.	—	—	55,32%	29,79%	31,48%	17,35%

M13	Variable name affects 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).	—	—	19,15%	12,77%	12,22%	12,33%

M14	Ignoring summation in 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.	—	—	44,68%	31,91%	26,67%	15,07%

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.	—	—	—	—	21,11%	8,22%

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.	—	—	—	—	11,11%	13,70%

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.	—	—	—	—	23,33%	12,33%

**Reported rates of misconceptions are based on students' performance in the Python programming language after the respective instructional treatments.*
Not all misconceptions appear in every phase, as their presence depends on the scope and design of the test instruments used.

CONCLUSIONS

No significant impact on misconceptions:

- the choice of a specific text-based programming language in introductory programming (Phase 1) [6],
- the application of program visualization techniques in reducing misconceptions (Phase 2) [7].

Significant impact on misconceptions:

- using a block-based programming language as a tool for *mediated transfer* after prior text-based instruction (Phase 3) [8],
- employing a *block-based programming language* for *mediated transfer* compared to traditional instruction (Phase 4) [9].

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.