


Python Programming Topics That Pose a Challenge for Students

Justyna Szydłowska ✉ 

University of Szczecin, Poland

Filip Miernik ✉ 

University of Szczecin, Poland

Marzena Sylwia Ignasiak ✉ 

University of Szczecin, Poland

Jakub Swacha ✉ 

University of Szczecin, Poland

Abstract

Learning programming is often considered as difficult, but it would be wrong to assume that all programming topics are equally tough to learn. In this paper, we make use of a gamified programming learning environment submission repository containing records of over 9000 attempts of solving Python exercises to identify topics which pose the largest challenge for students. By comparing students' effort and progress among sets of exercises assigned to respective topics, two topics emerged as especially difficult (Object-oriented programming and Classic algorithms). Also interesting are the identified differences between genders (indicating female students to fare better than male at the initial topics, and the opposite for the most advanced topics), and the scale of effort some students put to succeed with the most difficult exercises (sometimes solved only after tens of failed attempts).

2012 ACM Subject Classification Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases learning programming, programming exercises, gamified learning environment, learning Python

Digital Object Identifier 10.4230/OASICS.ICPEEC.2022.7

Funding The work described in this paper was achieved within two projects supported by the European Union's Erasmus Plus programme: the Framework for Gamified Programming Education (2018-1-PL01-KA203-050803) and FGPE Plus: Learning tools interoperability for gamified programming education (2020-1-PL01-KA226-HE-095786).

1 Introduction

Students that are trying to learn programming face a variety of problems, which concern many different fields. Obstacles and challenges encountered on the way relate to almost every aspect of the area. A question arises, which of them are the most difficult, or posing the largest challenge for students. Knowing that can be helpful in deciding where the focus of programming courses should be made, assigning sufficient time frames for learning respective topics, and selecting problems of adequate difficulty for the final exam.

In this paper, we investigate this matter on the case of the “Introduction to Python 3 programming” exercise set developed at University of Szczecin by a team led by the last co-author of this paper, as a part of the effort on the Framework for Gamified Programming Education project realized under the Erasmus+ international programme [6]. The exercise set covers all key aspects of Python, from the basics of the syntax to the Python implementation of classic algorithms, arranged in 12 thematic sections (called lessons). The exercise set is released as open source as a part of a larger collection, and can be freely reused and



© Justyna Szydłowska, Filip Miernik, Marzena Sylwia Ignasiak, and Jakub Swacha; licensed under Creative Commons License CC-BY 4.0

Third International Computer Programming Education Conference (ICPEEC 2022).

Editors: Alberto Simões and João Carlos Silva; Article No. 7; pp. 7:1–7:9

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

7:2 Python Programming Topics That Pose a Challenge for Students

modified [8]. What is important, the whole exercise set is gamified in order to constantly encourage students to keep learning. While solving the exercises, writing their own code, finding bugs and improving previous submissions, students receive various virtual rewards, and compete against each other for the top ranks in the leaderboard.

The exercises were provided on the FGPE PLE interactive learning web platform [13] to two groups of students (counting together 39 students, including 30 male and 9 female) attending introductory Python courses in the winter semester 2021/2022 at two business schools in Szczecin, Poland. FGPE PLE consequently records students' submissions at the server, including the result of their evaluation. During the whole semester, the students made over 9000 code submissions.

We decided to capitalize on the collected data to identify the programming topics (i.e., lessons, or thematic sections of the exercise set) which posed the largest challenge for the students, in belief that results obtained this way were more precise and reliable than those obtained by interviewing the students on the difficulty of respective topics – especially, if the latter is performed at the end of a course, when students' memory is already blurred.

In order to identify the topics which pose the most challenge to the students, two measures were employed:

- the average number of code submissions made for an exercise, which reflects how much effort was put by those who eventually solved an exercise,
- the lesson completion ratio, or the percentage of students who completed all exercises from a lesson out of all who started it, which shows for how many students the challenge was big enough to resign from completing a lesson.

The paper is organized as follows. In the subsequent section, we describe the context of our research and briefly review existing reports of similar kind. Next, we describe the structure and the contents of the exercise set, and present student progress data revealing which topics were the most challenging, and which were not. We then discuss the obtained results and conclude the paper.

2 Background and Related Work

Learning programming is difficult even to the extent of being called “IS student's worst nightmare” [21]. Having been diagnosed already a long time ago (see [1] and works cited therein), this problem has attained wide research interest ever since. One of the results of this interest were various attempts made to reduce difficulty of learning programming, by making use of, e.g., program execution visualization [2], automatic evaluation [9], gamification [13], or full-fledged educational games [5]. Despite all such efforts and improvements in educational process and technology, the problem continues to be significant as confirmed by a recent international survey [19].

An important research direction is focused at finding causes of this difficulty, which may be of various character [16]. Some researchers look for them within students, pointing to their prior knowledge [4] or individual patterns of learning [14]; some blame the teachers' own poor content knowledge [18] or the outdated teaching methods [7]; some indicate the differences in difficulty of learning various programming languages [17].

This paper contributes to yet another vein of research on programming learning difficulty, which aims at identifying the programming course topics that pose the largest challenge for students. The largest research of this kind that we are aware of was done on a cohort of around 1500 students at the Open University, United Kingdom. The data source was an online 'Python help forum' where students were recording and discussing encountered

challenges. The forum contained 178 discussions with a total of 1430 posts of which 29 were Python-related, 15 focusing on module-specific questions or issue reports and 19 on problem solving and general programming skills. Among the Python-related topics, IDE was the most frequently brought up (in 40 discussions), followed by Collections (in 21 discussions), Functions (16 discussions) and Error messages (15 discussions), whereas if we look at the median number of posts in a discussion, which may indicate the depth of a problem, the highest were noted for Functions (12) followed by Imports (11) and Error messages (10) [15].

The second largest study was done at Helsinki University of Technology, Finland, in a form of a survey administered to programming course participants in two subsequent years (2006–7). Data were collected from 459 students who passed the course and 119 who dropped out. Among the programming concepts, inheritance and abstract class, handling files, object-oriented concepts, and exceptions turned out to be the most difficult, while statements, loops and methods were the least difficult.

Another study in this vein was conducted in Chennai, India, involving 195 undergraduate students (56.41% female and 43.59% male) of Engineering and Technology who were taught Java Programming in the first semester of 2011. The most difficult topics to learn were noted to be “Concurrent programming”, “UI components with Swing” and “Generic programming”, whereas “Exceptions and Assertions”, “Event Handling” and “Interfaces and inner classes” were marked as moderately difficult; topics “Graphics programming”, “OO Access controls” and “Object Orientation” were perceived as less difficult, and “Fundamental programming structure in Java” as not to be difficult at all [20].

The results obtained from 145 students (120 female and 25 male) attending a one-semester introductory programming course at Buraimi University College, Oman, in 2013–14 indicate that the most difficult topics for students beginning their journey with programming are repetition structures, arrays and functions, the easiest being input/output statements, selection structure, parameters and operators [11].

Among 105 students (49.5% male and 50.5% female) of Sultan Idris Education University in Malaysia, the multidimensional array turned out to be the least understood, succeeded by looping statements, functions and the array data structure. Variables, constants and data types as well as selection statements were rated as mostly understood and input/output statements were best understood [3]. Another research involved 66 respondents, comprising programming students from the University of Dundee, and lecturers and teachers of programming from various universities in the UK, who were asked to indicate concepts and topics they found to be most difficult to cope with. The highest difficulty was given to copy constructors, operator overloading, templates, dynamic allocation of memory, pointers and other data structures (trees, linked-lists). In the middle, there were topics such as virtual functions, polymorphism, constructors/destructors, input/output and file handling, passing by reference/passing by value, and inheritance. Assessed as the easiest were looping operations, conditional operations, operators and precedence, basic function calling/program flow, and variable/function declarations [12].

A much smaller survey from central Anatolia, Turkey, involved 12 undergraduate students participating in the Internet Programming course in the fall of 2013. Among topics posing a challenge to learn, “syntax” was reported most often, followed by “functions and parameters”, “concepts, principles”, “assign variable” and “decision structures and loop” [22].

3 Challenging Topics in an Introductory Python course

3.1 Structure and Contents of the Exercise Set

The “Introduction to Python 3 programming” exercise set consists of 94 gamified exercises grouped in 12 consecutive lessons (see Table 1).

7:4 Python Programming Topics That Pose a Challenge for Students

■ **Table 1** Content of the “Introduction to Python 3 programming” course.

#	Lesson 1: Basics	Lesson 2: Strings	Lesson 3: Variables
1	Arithmetic operators	Strings	Creating variables
2	Nested parentheses	Apostrophes	Setting variable values
3	The order of arithmetic operators	Special characters	Modifying the value of a variable
4	Exponentiation	Raw strings	Setting several variables at once
5	Alternative number systems	Multi-line string literals	Changing the values of many variables
6	Real numbers and integers	String concatenation	Naming variables
7	Scientific notation	Strings vs. numbers	-
8	Division remainder	String repetitions	-
#	Lesson 4: Conditionals	Lesson 5: Loops	Lesson 6: Sets
1	Comparisons	Introduction to loops	Introduction to sets
2	Equality check	Ranges	Set initialization and modification
3	The else block	Loop counter	Loop over a set
4	Parity checking	Two parameters of the range function	Functions operating on set elements
5	Many cases	Three parameters of the range function	Subsets and operations on sets
6	Combining comparisons	The continue instruction	Turning a string into a set
7	Nesting comparisons	Nested loops	Operations on sets made from strings
8	-	Aggregates	-
9	-	Breaking a loop and the else block	-
#	Lesson 7: Lists	Lesson 8: String processing	Lesson 9: Dictionaries
1	Lists	ASCII codes	Dictionaries
2	Create and combine lists	String immutability	Accessing items in a dictionary
3	Extending a list	Searching for a substring	Dictionaries as a collection of keys
4	Shortening a list	Determining the substring position	Operations on dictionary elements
5	Checking the contents of a list	Replacing a substring	Operations on a whole dictionary
6	Accessing list items by index or value	Changing the letter case	Default values of dictionary elements
7	Inserting and deleting elements	Centering strings	Using a dictionary to translate words
8	List slicing	Splitting and combining strings	-
9	Counting items in a list	-	-
10	Converting strings or sets to lists	-	-
#	Lesson 10: Functions	Lesson 11: Object-oriented programming	Lesson 12: Classic algorithms
1	Defining a function	Introduction to classes	Binary search
2	Function result	Parameterized constructor	Quicksort
3	Function parameters	Displaying object contents	Fibonacci sequence
4	Parameter names	Defining a class	The problem of 8 queens
5	Default values of parameters	Inheritance	The knight’s tour problem
6	Variable number of parameters	Method overloading and base class access	The change-making problem
7	Unpacking parameters from the list	Accessing object’s class	-
8	Many results of the function	-	-
9	Local variables	-	-
10	Nested functions	-	-
11	Global variables	-	-

3.2 Topics’ Difficulty According to Students’ Effort and Progress

Figure 1 compares the two difficulty measures visually, with each dot representing an individual exercise, and its color denoting the lesson it belongs to. A reader is reminded that both the source of the presented data and the procedure of its processing were described in the Introduction.

As we can observe, generally, the further the lesson an exercise belongs to is placed in the set, the smaller is the share of students who solved it (note that the students were given an access to all exercises regardless of its topic from the very beginning, i.e., they were not required to complete the earlier lessons to access the later ones). Exercise Arithmetic operators was the one solved by most students (92.31%), whereas exercises The knight’s tour problem (2.56%), then The problem of 8 queens and The change-making problem (both

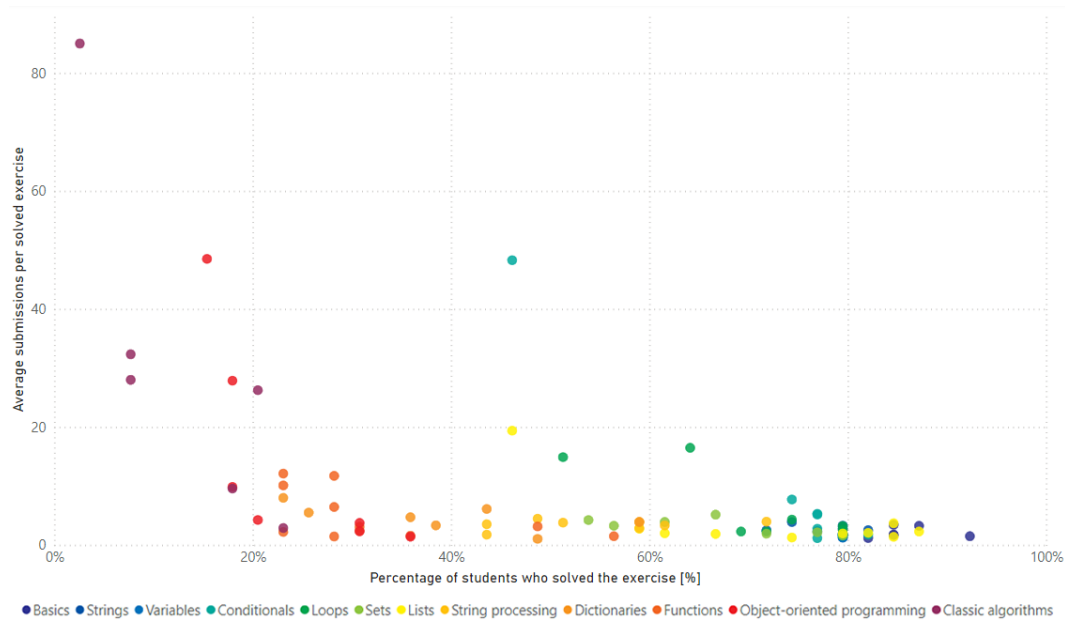


Figure 1 Average submissions per solved exercise by percentage of students who solved it.

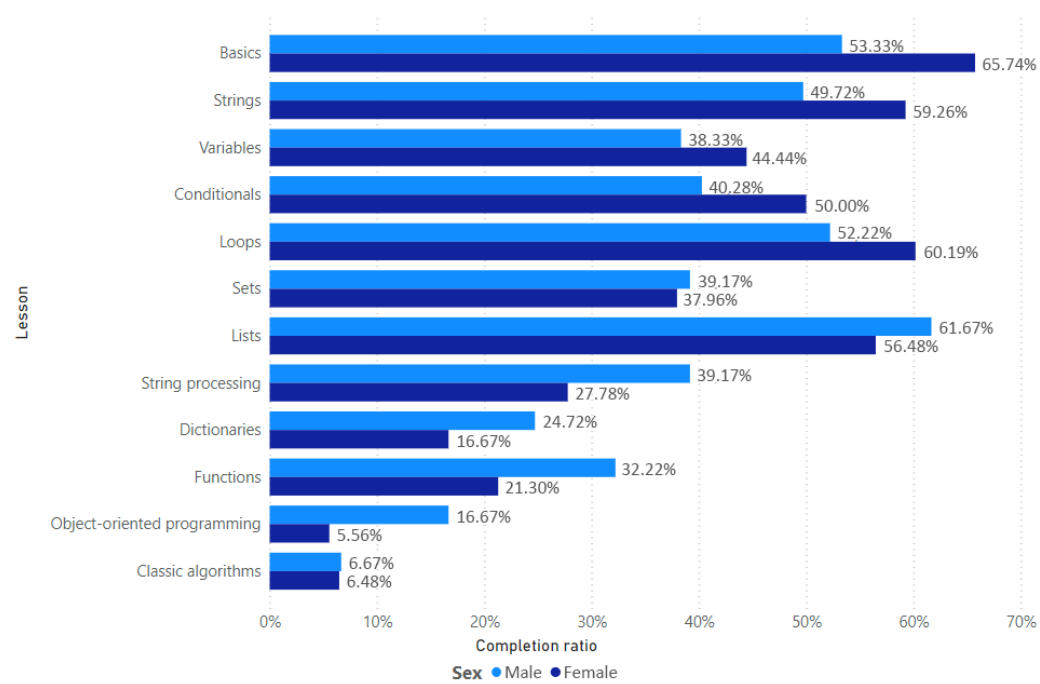


Figure 2 Completion ratio of every exercise (male and female).

7:6 Python Programming Topics That Pose a Challenge for Students

7.69%) were the ones solved by least students. As for the average number of tries students made to solve an exercise, the picture is more complicated: even in the later lessons, there are exercises solved in the first attempt by most students, and most exercises were solved after few submissions at most. However, we have some outliers: the most notable is exercise The knight's tour problem from lesson Classic algorithms which took on average over 80 tries to solve. The similar cases are exercises Defining a class from lesson Object-oriented programming and Nesting comparisons from lesson Conditionals which both took about 50 submissions on average.

In Figure 2, we present the average ratios of students who completed exercises from a given lesson, depending on their gender. For females, the average completion ratio ranged from about $\frac{2}{3}$ for the introductory first lesson to about $\frac{1}{18}$ for the Object-oriented programming lesson (which was the one before the last; interestingly, for the last topic, Classic algorithms, a bit higher ratio of about $\frac{1}{16}$ has been achieved among female students). To a lesser degree, females also struggled with two other lessons: Dictionaries (completion ratio of $\frac{1}{6}$) and Functions (completion ratio near $\frac{1}{5}$). For males, unexpectedly, the first lesson was not the easiest (completion ratio of about $\frac{1}{2}$), but the lesson on Lists (completion ratio of about $\frac{3}{5}$). Males struggled the most with the last lesson (Classic algorithms, completion ratio of about $\frac{1}{16}$), then with the topics of Object-oriented programming (completion ratio of $\frac{1}{6}$) and Dictionaries (completion ratio of about $\frac{1}{4}$).

In total, female students fared better in the first five lessons, and male students in the remaining seven. Such a result indicates that females more often than males grasped the concepts of programming from the very beginning, whereas males were better at dealing with more complex topics further on. Looking at the differences between the two genders, while for some lessons it was small, for some it was large. In particular, female students achieved a 23% higher completion ratio for lesson Basics, 19% for Strings, and 24% for Conditionals, whereas male students achieved a three times higher completion ratio for Object-oriented programming, 51% for Functions, 41% for String processing, and 48% for Dictionaries.

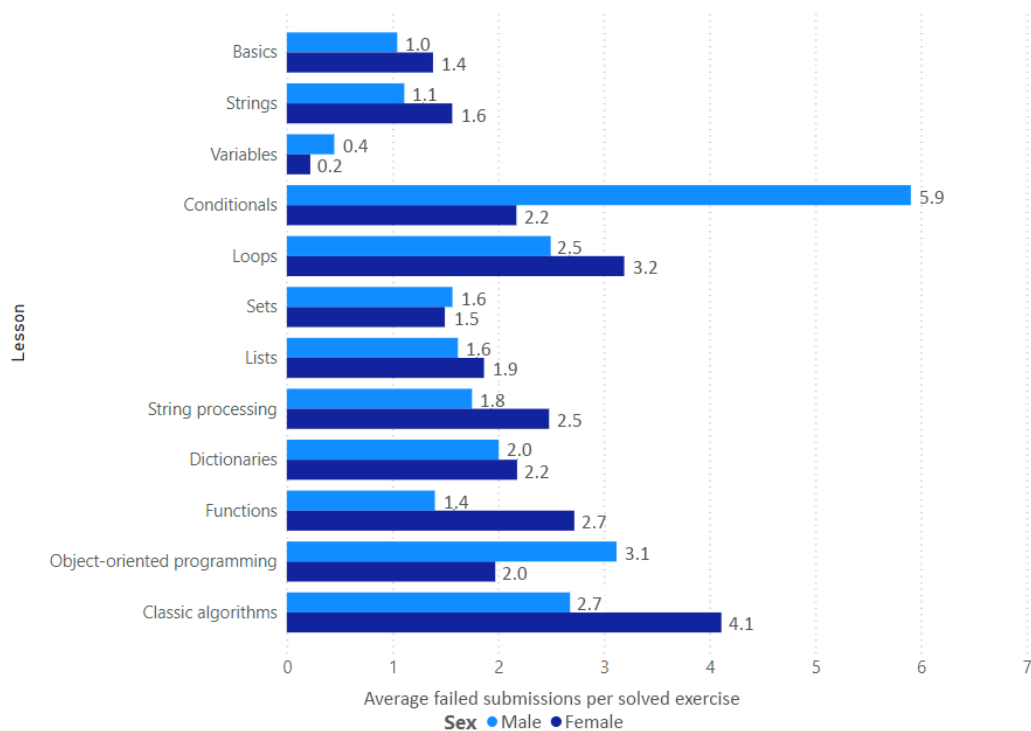
The number of failed submissions made by students who eventually solved an exercise averaged for every lesson and disaggregated by gender has been presented in Figure 3.

For females, the average number of failed submissions ranged from about 0.2 (the vast majority of submissions were accepted on the first try) for the Variables lesson to over 4 per exercise for the Classic algorithms lesson. Females have also sent a significantly high number of submissions in lesson Loops (3.2). For the remaining lessons, the average numbers ranged from 1.4 to 2.7. In the case of males, the average number of failed submissions ranged from about 0.4 for the Variables lesson to 5.9 for the Conditionals lesson. Male students also needed a small number of tries before passing in lessons Basics (1.0), Strings (1.1) and Functions (1.4). For the remaining lessons, the average numbers ranged from about 1 to 3.

The average number of failed submissions per solved exercise for both male and female students is very similar in most lessons. The greatest difference between the two genders can be seen for lesson Conditionals, where males have sent about 3.7 more failed submissions per exercise on average than females. The opposite situation happened for lessons: Classic algorithms (1.4 less submissions sent by male students on average), Functions (1.3 less), and Object-oriented programming (1.1 less).

4 Discussion

The prior work focused at surveys reporting students' own estimation of difficulty, whereas the results presented here were based on objective data measuring students' ability to solve particular exercises and the effort they made to accomplish that. While the latter also has



■ **Figure 3** Average failed submissions per solved exercise for every lesson (by gender).

its limitations – students may discontinue a course at any time for reasons other than its difficulty, and the number of failed submissions may be underestimated as students can develop and test their code in another environment, and paste it into the learning environment only after they believe it is correct – we still consider it a more reliable estimator of topic difficulty than students’ opinion, especially declared long after solving the exercises actually took place.

Comparing the presented results to those reported in prior work, no clean pattern emerges. Our results, similar to [12] and [10], point to object-oriented programming as one of the most difficult topics, and loops as one of the easier ones, whereas works [20] and [3] reported almost opposite results regarding these two topics. The latter study, however, agrees on syntax not posing a learning challenge; this is in contrast to [22], which on the other hand is consistent with our results with regard to considering functions as difficult, alike also [11]. This may stem from differences in both programming languages taught and the educational context.

The reviewed literature did not give sufficient attention to gender differences. The results of our study show that female students fare better at handling the basics of programming, but struggle at later topics, whereas those male students who pass the early barriers, are more capable of passing the exercises belonging to more difficult topics as well.

5 Conclusion

In this paper, we extended the knowledge of which programming learning topics pose the largest challenge for students with new results obtained from two groups of students playing

with a set of gamified Python exercises.

Unlike the most previous research in this vein, instead of surveying the students about the perceived difficulty of respective topics, we based our analysis on objective data comprising the share of students who completed all exercises belonging to a lesson on a given topic, and the average number of submissions the students made before their solution was accepted.

While, in general, the measured difficulty increases the further in the exercise set we proceed, the most notable observation is the very sharp rise of the difficulty for the two final lessons, devoted to Object-oriented programming and Classic algorithms, respectively. Possibly, these two topics should not belong to an introductory programming exercise set.

By distinguishing the gender of students, we were able to reveal in our data that female students had an easier start with learning programming, whereas the male students were more inclined to continue solving exercises till the end, even in spite of numerous failed attempts.

A very interesting observation was made thanks to measuring the number of submissions before one was accepted: for some exercises, a number of students kept trying and eventually succeeded even after failing tens of times. We link this observation with the fact the exercises were gamified, which most probably helped to keep the engagement and motivation high for at least a part of the students. Proving that, however, would require a comparison with a group learning with non-gamified exercises. This indicates the direction of our future work.

References

- 1 Yorah Bosse and Marco Aurélio Gerosa. Why is programming so difficult to learn? patterns of difficulties related to programming learning mid-stage. *SIGSOFT Softw. Eng. Notes*, 41(6):1–6, 2017. doi:10.1145/3011286.3011301.
- 2 Michael P. Bruce-Lockhart and Theodore S. Norvell. Lifting the hood of the computer: program animation with the teaching machine. In *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No.00TH8492)*, volume 2, pages 831–835, 2000. doi:10.1109/CCECE.2000.849582.
- 3 SitiRosminah MD Derus and Ahmad Zamzuri. Difficulties in learning programming: Views of students. In *Proc. 1st International Conference on Current Issues in Education*, pages 74–78, Yogyakarta, Indonesia, 2019. doi:10.13140/2.1.1055.7441.
- 4 Dimitrios Doukakis, Maria Grigoriadou, and Grammatiki Tsaganou. Understanding the programming variable concept with animated interactive analogies. In *Proceedings of the The 8th Hellenic European Research on Computer Mathematics & Its Applications Conference (HERCMA'07)*, 2007. URL: http://users.sch.gr/adamopou/docs/syn_HERCMA2007_doukakis.pdf.
- 5 Michael Eagle and Tiffany Barnes. Experimental evaluation of an educational game for improved learning in introductory computing. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education, SIGCSE '09*, pages 321–325, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1508865.1508980.
- 6 FGPE Consortium. Framework for Gamified Programming Education, 2020. accessed on 22 April 2022. URL: <http://fgpe.usz.edu.pl>.
- 7 Mark Guzdial and Elliot Soloway. Teaching the nintendo generation to program. *Commun. ACM*, 45(4):17–21, 2002. doi:10.1145/505248.505261.
- 8 Jakub Swacha, Thomas Naprawski, Ricardo Queirós, José Carlos Paiva, José Paulo Leal, Ciro Giuseppe De Vita, Gennaro Mellone, Raffaele Montella, Davor Ljubenkov, Sokol Kosta. Open Source Collection of Gamified Programming Exercises. In *Proceedings of the thirty-seventh Information Systems Education Conference ISECON 2021*, pages 120–123, Oak Creek, 2021. Foundation for IT education. URL: <http://proceedings.isecon.org/download/co8h5jrbkvipjznly7dp>.

- 9 Mike Joy, Nathan Griffiths, and Russell Boyatt. The Boss online submission and assessment system. *J. Educ. Resour. Comput.*, 5(3):2–es, 2005. doi:10.1145/1163405.1163407.
- 10 Päivi Kinnunen. *Challenges of teaching and studying programming at a university of technology – viewpoints of students, teachers and the university*. Doctoral thesis, Helsinki University of Technology, Espoo, Finland, 2009. URL: <https://aaltodoc.aalto.fi/handle/123456789/4710>.
- 11 Sohail Iqbal Malik and Jo Coldwell-Neilson. A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 22(3), 2017. doi:10.1007/s10639-016-9474-0.
- 12 Iain Milne and Glenn Rowe. Difficulties in learning and teaching programming – views of students and tutors. *Education and Information Technologies*, 7(1):55–66, 2002. doi:10.1023/A:1015362608943.
- 13 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, Jakub Swacha, and Filip Miernik. An open-source gamified programming learning environment. In *Second International Computer Programming Education Conference (ICPEC 2021)*, volume 91 of *OASICS*, pages 5.1–5.8, Saarbrücken, Wadern, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.ICPEC.2021.5.
- 14 D. N. Perkins, Chris Hancock, Renee Hobbs, Fay Martin, and Rebecca Simmons. Conditions of learning in novice programmers. *Journal of Educational Computing Research*, 2(1):37–55, 1986. doi:10.2190/GUJT-JCBJ-Q6QU-Q9PL.
- 15 Paul Piwek and Simon Savage. Challenges with learning to program and problem solve: An analysis of student online discussions. In *SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 494–499, New York, 2020. ACM. URL: <http://oro.open.ac.uk/68074/>.
- 16 Yizhou Qian and James Lehman. Students’ misconceptions and other difficulties in introductory programming: A literature review. *ACM Trans. Comput. Educ.*, 18(1), 2017. doi:10.1145/3077618.
- 17 Łukasz Radliński and Jakub Swacha. C# or Java? – analysis of student preferences. *Studies & Proceedings of Polish Association for Knowledge Management*, 58:101–113, 2012.
- 18 Philip Sadler, Gerhard Sonnert, Harold Coyle, Nancy Cook-Smith, Jaimie Miller-Friedmann, and Harvard-Smithsonian Center. The influence of teachers’ knowledge on student learning in middle school physical science classrooms. *American Educational Research Journal*, 50(5):1020–1049, 2013. doi:10.3102/0002831213477680.
- 19 Simon, Andrew Luxton-Reilly, Vangel V. Ajanovski, Eric Fouh, Christabel Gonsalvez, Juho Leinonen, Jack Parkinson, Matthew Poole, and Neena Thota. Pass Rates in Introductory Programming and in other STEM Disciplines. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, pages 53–71, Aberdeen, 2019. ACM. doi:10.1145/3344429.3372502.
- 20 M. Sivasakthi and R. Rajendran. Learning difficulties of ‘object-oriented programming paradigm using Java’: students’ perspective. *Indian Journal of Science and Technology*, 4(8):983–985, 2011. doi:10.17485/ijst/2011/v4i8.9.
- 21 Amy B. Woszczyński, Hisham M. Haddad, and Anita F. Zgambo. An IS student’s worst nightmare: Programming courses. In *SAIS 2005 Proceedings*, 2005. URL: <https://aisel.aisnet.org/sais2005/24/>.
- 22 Büşra Özmen and Arif Altun. Undergraduate Students’ Experiences in Programming: Difficulties and Obstacles. *Turkish Online Journal of Qualitative Inquiry*, 5(3):9–27, 2014. doi:10.17569/tojqi.20328.