Supporting Gender Equality in Computer Science Through Pre-Introductory Programming Courses

András Margitay-Becht[1,2] and Udayan Das[1]

[1] Saint Mary's College of California, 1928 St. Mary's Road, Moraga, CA 94575, USA
[2] Eötvös Lóránd University, Pázmány Péter sétány. 1/C, 1117, Budapest, Hungary
abecht@inf.elte.hu

Abstract. The past thirty years have seen an exponential growth in the presence of computers in everyday life. However, some studies find that introductory programming courses in university curricula might reduce students' willingness to engage with the material further. To complicate matters, despite extensive efforts of the past decades, there are still corners of the computer science landscape that are unwelcoming to female practitioners, implicitly or explicitly discouraging female students from the profession.
This paper discusses an experimental university-level pre-introductory course targeting non-computer science students with limited or no background in programming. The explicitly stated goal of the course was to reduce trepidation regarding programming. The course saw equal participation from male and female students. The results were positive: students reported a reduced perception of difficulty regarding programming and an increase in the subjective importance of the area. Even more encouragingly more female students experienced a reduction in the perception of programming difficulty than male students.
The paper will discuss the structure of the course, the unique set of approaches that led to success, the lessons learned, and the new iteration of the class we are going to offer the upcoming year.

Keywords: Gender equality · Programming education · Computational thinking.

1 Introduction

There is a global shortage of programmers and computer science professionals, in spite of the relatively high salary. There are many reasons for this, but a contributor is the low and decreasing engagement of females with the discipline. In the mid 80-s, 37% of computer science students were female [1]. This shrank to 27% by 1997, then to 20.7% by 2006, and hit 18.7% in 2016 [2]. The total number nearly doubled be-tween 1997 and 2016, from 6900 to 12,200, but this growth is significantly lower than the growth experienced among male students.

This paper will discuss why this discrepancy might exist, and what are some possible ways of reducing this effect. We will also introduce our approach that targeted the fear of computer science in general, but ended up also empowering female students in the pursuit of computer science knowledge.

J.-P. Pellet and G. Parriaux (Eds.): ISSEP 2023 Local Proceedings, pp. 37–48, 2023.
https://doi.org/10.5281/zenodo.8431898

2 The Problem with Introductory Classes

One of the problems of teaching computer science, whether for male or female students, is the fact that the topic is generally considered scary (for a few examples see [3–5]). Of particular interest is the research done by Alford et al in 2017[6]. The original purpose of their research was to investigate the gender gap, but they found that students in general, irrespective of gender start out intimidated by programming. The student population they investigated were mostly engineering students who had to take an introductory programming class (so many of them were there because they had to and not because they wanted to). The total number of students investigated were close to 2000, of whom about 35% were female. Importantly, they found no real difference between the interest of male and female students before taking the class. When asked what grade they were expecting, male and female students again responded similarly, but to a question regarding how confident they were that they can successfully complete the course, the male students responded more positively, in alignment with previous findings. Interestingly, when asked how confident the stu-dents were in their ability to succeed in the course, the pre-test results showed greater confidence for men than women, but the post-test saw men drop in their self-assessment more than women (indeed, in some samples women increased their con-fidence throughout the course). The greatest takeaway for us was hidden in the phrasing of the previous sentence: men and women both left the introductory classes less interested and confident and more intimidated than at entry time. This is in agreement with LaBouliere's findings from 2015: middle school girls increased in their programming understanding and confidence, but greatly decreased in their inter-est in programming[7]. Similarly, Rubio reports a loss of interest for female students as well[1].

It appears that introductory courses might not be the best places to provide an in-spiring experience for students. These classes have to discuss a significant curriculum of complex topics, covering those usually takes up the whole course, and there is little time to slow down and spend some time working on an interesting project or real life problem. Introductory classes are excellent for students who are already interested in the subject and determined to follow through. Indeed, they are designed to serve this audience, so the speed of the classes and the amount of material they cover are scaled to support these students. But this leads to the above situation where students who are not already positively predisposed towards the discipline might be further turned off by the introductory classes.

A solution is to start with a pre-introductory experience, like the one discussed in [8]. As Schindler and Muller state: "The first contact with programming is crucial to keep students in the long run", so if the curriculum design or educational system allows for it, creating a "first contact" experience might be valuable.

Our approach [9] is similar: we are trying to provide an educational experience that is primarily focusing on showcasing the value, and indeed the joy of programming, while covering some basic concepts of coding. We have created our class, “Coding is Fun”, not as a replacement for introductory classes, but as a place for curious students to explore their curiosity, see if they find value and joy in programming, in order to provide them with the mindset that will set them up for success in the introductory programming class. Our approach differs from the above cited Schindler and Müller approach by utilizing the advantage presented by the fact that we are teaching at a small liberal arts university with class sizes around 20, allowing us to give control of a large portion of our class over to our students. Instead of teachers, we became guides, and as we guided them throughout the basics of programming along their own interests, pivoting as they wanted, we succeeded in increasing their interest and reducing their apprehension. It also allowed us to use the approachable Scratch programming environment, something that would have been unsuitable for an introductory class at university level. Our course also serves as an optional course unattached to any program, so it is accessible not only to computer science students, but for anyone interested in general.

We have designed this experience for all students, irrespective of gender. We have found, however, that our female students seem to have benefitted more than our male students from this approach. In this paper we will quickly discuss some of the issues and prejudices affecting female students of computer science, then discuss our course and the outcomes of the pre-introductory experience.

3 Key Issues for Female Students

3.1 Skills

First and foremost it is important to discuss whether there is a capability gap between male and female computer science students. Murphy et. al., investigating the gender differences among male and female students at a computer science program in 2006 found that women in general started the program knowing less than men did, but as they progressed through the program, both their knowledge and academic performance (GPA) caught up to that of the male students. In 2015 Akinola completed an empirical study of programming skills between male and female groups of two and four, finding no statistically relevant difference between the genders based on either efficiency or accuracy [10]. Akinola concluded that the underrepresentation might come from different interests or fear, lack of confidence.

In a 2018 study Kallia and Sentance compared mostly 11th grade students from 7 different UK schools, finding that boys and girls performed roughly the same [11]. A 2019 study at the Graz University of Technology found similar results: freshmen female computer science students performed just as well as males did[8].

Study after study confirms our personal experiences, that female students are not worse at programming and computer science related tasks and courses than their male counterparts. The underrepresentation must come from another source.

3.2 Interest

A frequent component of research projects focusing on the gender gap is the apparent lower interest of female students in the field. Pau et. al. reports on the generally held preconception that bad experiences with programming classes can alienate female students from pursuing the discipline [12]. They found that with proper support and structure, programming courses can be empowering to female students.

Funke et. al. surveyed 63 Bavarian computer science teachers [13]. Most of them reported no dissimilarities between boys and girls, only three categories showed meaningful differences: girls were perceived as more structured but less confident and interested. Master et. al. report on an experiment enriching the computer science experience of 6-year-old students with the use of robotics[14]. Like Funke, they also found that boys had greater intrinsic interest than girls, but the introduction of robotics increased girls' interest significantly more than boys', drastically reducing the interest gap between the two groups. Braga and Motti explored a similar age group, 7-10 year olds [15]. The children were invited to participate in programming exercises inspired by the worlds of Frozen, Minecraft and Angry Birds. They found that if the environment is engaging enough, both girls and boys are equally likely to participate. Moreover, "girly" themed experiences – as the article calls them – were not a major motivational factor for the girls.

3.3 Confidence

Female students, overall, tend to be less confident than their male counterparts, often undervaluing their own capabilities. This was the central question of Kallia and Sentence's research, finding that despite similar performance, girls regularly underestimate themselves [11]. More worryingly, girls also scored lower on self-efficacy than boys, indicating need of better support. Similar results were found in [8, 13] as well: female students matching the performance of male students, but rate themselves worse than males do.

3.4 Possible Solutions

If lower confidence and motivation keep highly competent girls and women from pursuing computer science, it is important to address these issues directly. Rubio et. al. suggested creating separate introductory tracks for people with different back-grounds, enabling easier transition into the discipline [1]. Alternatively, they suggested more contextualized classes like Media computation, robotics, or animation, where the utility of the field is easier to experience. Their own approach focused on using physical computing, which led to a reduced failure rate for females in the class, although it still caused a reduction in interest in computing among female students, like the alternative programming approach did.

Pau et. al. discuss the key issues they found that can increase the positive experience of an introductory programming class for female students [12]:

1. Programming tasks that are connected to real-life issues and actual problem solving are a lot more engaging, similar to Rubio's recommendations above
2. When time pressure is removed and students are allowed to work from home, they find the class a lot more beneficial
3. Parental support from home
4. Higher mathematics performance helps transitioning into programming

We have incorporated some of these findings into our course design to make our class more appealing – to female and male students alike.

4 Pre-Introductory Programming: a Combined Solution

4.1 Goals of the Course

The primary purpose of our course, called "Coding is FUN" was to show that, as the title describes, coding can be fun. At our university, we offer a mandatory January Term experience for the students. The instructors are encouraged to create unusual and experimental courses, and we designed this class to provide an opportunity for students who have not yet tried their hands at programming to explore the area a little bit. The course description itself that was available to the students before enrol-ling explicitly stated this.

4.2 Recruiting and Student Population

The course was originally intended for freshmen students just entering college, to help them learn about potential major opportunities like Computer Science (CS) or Data Science (DS). In the end, we ended up with a significantly more diverse group spanning from freshmen to seniors. The recruitment of students for the course was largely based on word-of-mouth. The university's Tech Club overseen by one of the instructors held two short super-introductory programming workshops. The upcoming course was announced at these workshops. Two of the students involved in these workshops also ended up becoming peer tutors. Instructors also shared the course information with peers in other departments. Many students in the CS program and DS programs shared this upcoming course with roommates and friends. One CS senior for example had 2 roommates attending the course.

Since our university is a small liberal arts college, class sizes are traditionally be-tween 15 and 25 students. We were initially worried that there might not be enough interest for the course, but in the end we ended up with 23, close to the maximum class size allowed.

A point of pride is the diversity of students we managed to address, both by gender and by major of study. We had 7 students of Business-adjacent majors, 4 students of varied STEM majors, 4 Psychology majors and 4 liberal arts majors in the class, in addition to the 4 freshmen who did not yet have declared a major. The gender breakdown ended up being 9 female and 14 male students completing the class.

4.3 Structure of the Class

The course was offered every Wednesday, 2.5 hours per meeting, during the January of 2023. The class was delivered in a synchronous online format, allowing students the ability to work from home. To reduce the impact of time pressure further, the course required the students to create only a single program as their final project; no tests, quizzes or homeworks were assigned. They were also allowed to work on their project whenever they wanted, with ample support being provided for them both online and offline.

The intended layout of the course was a session introducing Scratch programming, then a session of more advanced programming topics based on student choice, then a session on design and computational thinking and a final session for the students to showcase their content. Due to great student interest, this plan expanded by 50% to provide them with the extra opportunities they asked for.

We decided to use Scratch as the initial introductory language as it is extremely approachable even for 5-year-olds, and it all but removes any concern about syntax in programming. It is also complex enough that the basics of imperative programming can be found in it: variables, conditional statements, loops, functions/methods and lists, all in an approachable, graphical environment. It is also built around events, so students also learn event-based control. Since the class aimed to be a pre-introductory course to be followed up by a traditional introductory course, the short-comings of the programming language were considered less important than the immense ease-of-access benefit the language provided. This was a popular choice among students, and most of them stuck to using Scratch throughout the class.

During the first class session we set up a Slack channel to improve communication with the students. After the first class, a poll was posted there, asking if they want more Scratch practice during the second class, or want to see the basics of Spread-sheet programming or Python. To our surprise, while Visual-Basic based spreadsheet programming was entirely unpopular, many students showed interest in both more Scratch and learning some Python, so we decided to pivot the course plan: during the normal class time a Scratch session took place, and an optional Python period was added in the afternoon. Highlighting student interest, this optional period was well attended, just like a second optional Python session, that provided further details on the language. This was great feedback for us, as students seemed to have wanted to engage with the material significantly more than we expected – an insight that will be incorporated into the next iteration of the class.

During the third and fourth weeks additional scaffolding was provided to the students in the form of consultation periods both synchronously and asynchronously over Slack or e-mail. The third class period focused on how the design and computational thinking principles can be utilized to come up with a final project, and iteratively design, implement and test it. During the final class period the students showcased projects far more complicated than we expected or imagined, demonstrating that they have spent a significant amount of time outside the classroom learning additional techniques just for the amusement of themselves.

4.4 Scaffoding and Support

Student support is a critical element for bridging-the-gap for students who may not have as much exposure to programming or programming-adjacent materials in the past. Students need to know that programming is not a magical thing that some people just get and others cannot, but that programming is a skill that can be learned and honed. As with many other human endeavors and activities, some people are naturally more adept at programming, but that does not mean that others are incapable of learning this skill. The presence of in-class support provides students with the opportunity to talk through some difficulties, particularly for those students who are hesitant to talk to the class as a whole. Peer tutoring programs have demonstrated success, particularly in Computer Science learning contexts and can improve retention and student achievement [16].

There were 3 students who served as peer tutors for the course. They were selected such that they would be from different levels of proficiency. One student had recently completed Programming I (Programming with Python), one student had recently completed Programming II (Data Structures and Algorithms with Python), and the other student would be graduating with a Bachelor's Degree in Computer Science in 6 months. Thus, one student was at a first year level, the second student was 2nd year level, and the final student was at a senior (4th) year level. The different levels are beneficial in peer tutors to express to participants that learning programming is a journey and people at all levels have something to share. Peer tutors at different levels also bring different benefits with those fresh out of an introductory programming course having had recently experienced their own journey from insecure to self-reliant confident when it comes to programming, at the other end the senior level student brings the benefit of having more experience and knowledge yet being closer to the students than the instructors. Peer tutors always have the benefit of allowing more informal exchanges between students and peer tutors. The tutors for this class were also relatively diverse with 1 woman and 2 individuals from minority groups.

Support in this course involved both during class session support (peer tutors in groups) as well as peer-tutor led sessions during the project phase. About 8 students took advantage of sessions to work on the project during an open lab session with a peer tutor. In the week preceding the final presentations (week 3 of the course) these sessions were held. Instructors also held support sessions during this period with an additional 2 students attending. Altogether a total of 10 students used the support sessions. The availability of multiple sessions at different times of day and with dif-ferent individuals surely contributed to many students taking advantage of the ses-sions. Slack was also used as an online forum space for communication, community, and support and students took advantage of Slack to troubleshoot code snippets, and importantly, share their work with all other students at the end. Further along in the CS program students commonly submit work to GitHub and so this kind of shar-ing is possible. But starting this kind of sharing enables students to learn from each other, and in the instructors' experience get impressed and inspired by each other. No amount of code samples provided by the instructors can match the value of seeing peer work.

5 Methodology and Results

Grading and assessment were done completely separately in this course. For grading, we used a growth mentality. As this course was an elective class that did not count for any major field of study or the core curriculum of the university, and only carried a minuscule credit value (less than 0.7% of the total credits required for graduation), we felt comfortable assigning grades based on the improvements our students made in the small amount of time available in the class. Our key assessment focused on the student experience in the course and the change of attitude we were hoping to achieve. To measure this, we created a pre-class and a post-class questionnaire for the students, focusing on their experience with and attitudes towards programming. Two key questions are worth discussing in more detail: the perception of difficulty and the perceived utility of programming. The following tables will show information from the 15 of the 23 students who filled out both surveys. It is interesting to note, that close to 90% of the female students filled out both the pre- and post-class questionnaires, while only 50% of male students did so.

5.1 Perceptions of Difficulty

To test the perceptions of difficulty of programming, in the pre-test we asked the students how hard the class will be for them. Not surprisingly, none of them said they expected the course to be too hard for them, as it was an elective course working with a positively biased audience. Most of the students, however, chose the tentative "I can probably do it" option, with only 3 (2 males and 1 female) selecting the assertive "I can for sure do it".

At the end of the class, we were happy to find that close to half of the respondents found programming at least somewhat easier than they expected. Most of the rest found that programming was about as hard as they were expecting, with 2 students (a male and a female) feeling that programming was a bit harder than they thought initially.

While the above results are already exciting, they become even more so if we look at the gender breakdown of the responders. Table 1 shows the breakdown of the responses: the rows contain the questions from the pre-class questionnaire, while the columns the options from the post-class questionnaire. In the cells the values are in the form of [male : female] students giving that combination of answers. In alignment with the established literature, we found that six of the eight students who found programming to be easier than expected were females, while only four of the six who found it as hard as they expected were males. This means that the majority of the perception gain happened to the female students in the class, empowering and encouraging them to pursue programming further.

Table 1. Perceptions of programming difficulty before and after the course. Rows contain the post-course feedback, columns the pre-course feedback. Results are reported as [males : females]

	Too hard for me	I can probably do it	I can for sure do it
A bit harder than expected	0:0	1:1	0:0
About as hard as I expected	0:0	3:2	1:0
Easier than expected	0:0	1:4	0:1
A lot easier than expected	0:0	0:0	1:0

5.2 Perceived Utility

While it is exciting to see that the class increased the confidence and reduced the worry of female students, that alone is not going to improve participation if they find programming to be unimportant. To measure the perceived importance of programming, the pre-class questionnaire asked the students' estimation of the likelihood that programming will be useful for them personally. We had two students (both female) say that they did not expect to use programming at all, five students (4 males, 1 female) say that they will definitely use programming, the rest fell into the more tentative maybe category.

After the class, we were excited to find that six out of the eight females thought that programming will be more useful to them than they thought before the class, and one of the remaining two already thought it was going to be useful and found confirmation in her experience. The male students had a lot more varied journey: one student said that programming will be less useful than they initially expected, and all three of the students who were expecting programming to be a lot more useful to them were also males. It was also exciting to us that the class seems to have rein-forced initial positive expectations: all 5 students who were certain of the usefulness of programming experienced increased valuation of it.

Similar to above, Table 2 showcases the breakdown of [male:female] respondents based on their preclass and post-class responses.

Table 2. Perceptions of the utility of programming before and after the course. Rows contain the post-course feedback, columns the pre-course feedback. Results are reported as [males : females]

	I won't use it	I might use it	I will use it
Less useful than expected	0:0	1:0	0:0
About as useful as expected	0:1	0:1	0:0
Somewhat more useful than expected	0:1	1:4	2:1
A lot more useful than expected	0:0	1:0	2:0

5.3 Overall Results

Like the above cited literature, we have found no difference in the quality of work done by male and female students. Indeed, the most complex project was created by a female student, and every female student turned in a project that greatly exceeded the expectations – and the material covered in the class. This is an excellent indicator of increased self-efficacy, that was demonstrated by nearly all students. We hope that this skill will prove to be transferable to other areas as well.

Computational thinking, especially decomposition, was clearly demonstrated in most projects. This will reinforce and expand the students' critical thinking skills. We have seen both explicit and implicit use of design thinking. The most impressive ex-ample was one student who worked together with her younger brother to create a game with him for him to enjoy. This was both a touching moment – siblings using a class exercise to socialize remotely over Zoom – and also a great application of design thinking principles, working with and for a "client" to create a desired product. A final indicator of the success of the class, that out of the 15 responders, 13 re-ported desire to continue learning programming, either on their own, or in some kind of structured manner.

6 Next Steps

Due to the great success of the class, it will live on in the January of 2024, with some modifications. The greatest one of these will be an expansion to a full class, increasing the meeting times from four to 15. The increase in contact hours can provide an opportunity for introducing pair programming in the class. Pair programming and peer programming are also known to be effective learning techniques within software engineering and are a significant element of Agile Software Development practices. Talking through the thought process behind code development greatly strengthens students' overall programming skills. Pair programming has been demonstrated to improve retention and student confidence [17].

A change in recruiting efforts will attempt to address the needs of incoming fresh-men. The class will continue to target students with no programming background, but it could serve as an ideal first experience not only for students who are programming curious, but also for those who know they want to pursue a career in computer or data science and want a more gradual on-ramp to programming prior to taking their introductory course.

The course will also aim for a more layered outcome regarding further study for the students: aside from continuing to study on their own or just taking an introductory class, they might be able to pursue a certificate in programming or website development, a minor in programming or data science, or even a major in computer science or/and data science. The majors and minors at our institutions are created in a way that they scale easily. We consider computer and programming abilities necessary for the enlightened citizens of 2023, and this course might provide a gateway for students of all backgrounds and interests to expand their education with some 21st century skills.

7 Conclusion

Optional pre-introductory experiences implement separate introductory tracks for students with different backgrounds. Those who enter university with some back-ground in programming can enroll directly into an introductory programming class. Those, however, who are worried about their own skills or even underestimate their own abilities, as many female students seem to, can participate in a pre-introductory course to alleviate their concerns and improve their self-confidence. A student-centric, real-life grounded pre-introductory class can showcase the usefulness and fun of programming, creating and reinforcing interest. For example, a student who enjoys drawing can use programming to create animations. A student interested in child psychology can use programming to create research tools to engage children – or analyze the results of the engagement. A biochemist can use programming to model molecules or analyze experimental data, an economist can model the success of a product or the trajectory of a nation. By being able to focus on inspiring students, these classes can serve as the affective counterparts to the cognitive focused introductory courses[18]. And by focusing on increasing student interest, breaking down barriers, improving self-efficacy and confidence, pre-introductory courses can help support female students exactly in the ways they need support to be able to start a career in the IT field – or to expand their interest with technology.

References

1. Rubio, M.A., Romero-Zaliz, R., Mañoso, C., de Madrid, A.P.: Closing the gender gap in an introductory programming course. Computers & Education. 82, 409–420 (2015). https://doi.org/https://doi.org/10.1016/j.compedu.2014.12.003
2. NSF - National Science Foundation: Women, Minorities, and Persons with Disabilities in Science and Engineering: 2019, https://ncses.nsf.gov/pubs/nsf19304/digest. Last accessed 2023/06/02.
3. Connolly, C., Murphy, E., Moore, S.: Programming Anxiety Amongst Computing Stu-dents—A Key in the Retention Debate? IEEE Trans. Educ. 52, 52–56 (2009). https://doi.org/https://doi.org/10.1109/TE.2008.917193.
4. Höök, L.J., Eckerdal, A.: On the Bimodality in an Introductory Programming Course: An Analysis of Student Performance Factors. In: 2015 International Conference on Learning and Teaching in Computing and Engineering (2015).
5. Wyeld, T., Nakayama, M.: Visualising the Code-in-Action Helps Students Learn Programming Skills. In: 2018 22nd International Conference Information Visualisation (IV). pp. 182–187 (2018). https://doi.org/https://doi.org/10.1109/iV.2018.00040.
6. Alford, L., Dorf, M.L., Bertacco, V.: Student Perceptions of Their Abilities and Learning En-vironment in Large Introductory Computer Programming Courses. In: 2017 ASEE Annual Conference & Exposition Proceedings. p. 28867. ASEE

Conferences, Columbus, Ohio (2017). https://doi.org/https://doi.org/10.18260/1-2–28867.

7. LaBouliere, J.J., Pelloth, A., Lu, C.-L., Ng, J.: An exploration of the attitudes of young girls toward the field of computer science. In: 2015 IEEE Frontiers in Education Conference (FIE). pp. 1–6 (2015). https://doi.org/https://doi.org/10.1109/FIE.2015.7344265.
8. Schindler, C., Müller, M.: Gender gap? a snapshot of a bachelor computer science course at Graz University of Technology. In: Proceedings of the 13th European Conference on Soft-ware Architecture - Volume 2. pp. 100–104. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/https://doi.org/10.1145/3344948.3344969.
9. Margitay-Becht, A., Das, U.: Enhancing student learning through hidden motivational learn-ing outcomes. In: Enomoto, K., Wagner, R., and Nygaard, C. (eds.) Enhancing student learn-ing outcomes in higher education. Libri Publishing Ltd. (2023).
10. Akinola, S.O.: Computer programming skill and gender difference: An empirical study. American journal of scientific and industrial research **7**(1), 1–9 (2015).
11. Kallia, M., Sentance, S.: Are boys more confident than girls? the role of calibration and students' self-efficacy in programming tasks and computer science. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education. pp. 1–4. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/https://doi.org/10.1145/3265757.3265773.
12. Pau, R., Hall, W., Grace, M., Woollard, J.: Female students' experiences of programming: it's not all bad! In: Proceedings of the 16th annual joint conference on Innovation and technology in computer science education. pp. 323–327. Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/https://doi.org/10.1145/1999747.1999837.
13. Funke, A., Berges, M., Mühling, A., Hubwieser, P.: Gender differences in programming: re-search results and teachers' perception. In: Proceedings of the 15th Koli Calling Conference on Computing Education Research. pp. 161–162. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/https://doi.org/10.1145/2828959.2828982.
14. Master, A., Cheryan, S., Moscatelli, A., Meltzoff, A.N.: Programming experience promotes higher STEM motivation among first-grade girls. Journal of Experimental Child Psychology. 160, 92–106 (2017). https://doi.org/https://doi.org/10.1016/j.jecp.2017.03.013.
15. Braga, C., Mochetti, K.: Programming teaching tools and the gender gap in the Information Technology field. In: Anais do Workshop de Informática na Escola. pp. 70–79. SBC (2018). https://doi.org/https://doi.org/10.5753/cbie.wie.2018.70.
16. Servin, C., Pagel, M., Webb, E.: An Authentic Peer-Led Team Learning Program for Community Colleges: A Recruitment, Retention, and Completion Instrument for Face-to-Face and Online Modality. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. pp. 736–742. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/https://doi.org/10.1145/3545945.3569851.
17. McDowell, C., Werner, L., Bullock, H.E., Fernald, J.: Pair programming improves student retention, confidence, and program quality. Commun. ACM. 49, 90–95 (2006). https://doi.org/https://doi.org/10.1145/1145287.1145293.
18. Bloom, B.S., Krathwohl, D.R.: Taxonomy of educational objectives: The classification of educational goals. Book 1, Cognitive domain. longman (1956).