frontiers | Frontiers in Computer Science

TYPE Curriculum, Instruction, and Pedagogy
PUBLISHED 12 December 2022
DOI 10.3389/fcomp.2022.983237

OPEN ACCESS

EDITED BY
Gerrit C. Van Der Veer,
University of Twente, Netherlands

REVIEWED BY
Eduard Enoiu,
Mälardalen University, Sweden
Sofia Papavlasopoulou,
Norwegian University of Science and Technology, Norway

*CORRESPONDENCE
Daniel Spikol
ds@di.ku.dk

SPECIALTY SECTION
This article was submitted to Digital Education, a section of the journal Frontiers in Computer Science

RECEIVED 30 June 2022
ACCEPTED 24 November 2022
PUBLISHED 12 December 2022

CITATION
Spikol D, Dybdal M and Elmeskov DC (2022) Student experiences in a university preparatory programming course. *Front. Comput. Sci.* 4:983237. doi: 10.3389/fcomp.2022.983237

Student experiences in a university preparatory programming course

Daniel Spikol*, Martin Dybdal and Dorte C. Elmeskov

Departments of Computer Science and Science Education, University of Copenhagen, Copenhagen, Denmark

Even though computer science is currently being integrated into primary and secondary education worldwide, we cannot yet make assumptions about our student's prior knowledge of computing. Every student might have different conceptions about the field of study they are about to enter. Students start at computer science programs with different prior experiences with programming, ranging from no experience to a high degree of proficiency. At the Department of Computer Science, University of Copenhagen, we have designed a 2-week voluntary summer kickstart course in programming to help students in this transition into our three computer science programs. To evaluate the course, we followed three groups of students. Group one with no/limited programming experience attended the kickstart course. The second group with no/little programming experience did not participate in the course, and the third group of students with programming experience did not participate in the class. We observed the kickstart course and conducted interviews. We followed up about 3 weeks after the start of the semester and then again at the end of the semester in December. Our findings suggest that the course reduces the gap in programming experiences and strengthens students' self-efficacy and sense of belonging. However, the approach creates a social gap between students who have not attended the course with no/limited experience at the beginning of the semester. Even though the students in December do not experience any difference between students who have attended the course and those who have not, it is important to consider this social gap at the beginning of the semester when designing and planning a preparatory course like the kickstart course.

KEYWORDS

computer science education, development/boot camp, Cs0 and Cs1 learners, education design research, problem-based learning, pedagogical approaches

1. Introduction

Computer science is currently being integrated into primary and secondary education worldwide, but we cannot yet make assumptions about our student's prior knowledge of computing. Every student might have different conceptions about the field of study they are about to enter. Students start at computer science programs with

different prior experiences with programming, ranging from no experience to a high degree of proficiency. Research on novice programmers and introductory programming courses (Robins, 2019; Von Hausswolff, 2022) has shown to be challenging for the students and generally have high failure rates. The field of computer science education has been focusing on these challenges. Yet, problems persist, and student engagement in computing is low compared to other studies (Bennedsen and Caspersen, 2007; Luxton-Reilly et al., 2018). Schulte and Knobelsdorf (2007) have highlighted that novice students need better problem-solving and design strategies.

General approaches for introductory programming courses generally focus on the tools, paradigms, programming languages, and environments (Kandemir et al., 2021). However, just focusing on the efficiency of teaching and learning programming proves problematic in introductory courses where there is a need to ensure a well-thought-through on-boarding process for new students to ensure a safe environment where they can understand each other and efficiently learn together, despite their interests and prior experience with the subject.

Therefore, at the Department of Computer Science, University of Copenhagen, we have designed a 2-week voluntary summer kickstart course in programming to help students in this transition into our three computer science programs. From surveys of students' prior experience, we know that the majority of the students starting on our bachelor programmes have little or no previous programming experience (approximately 60%). We have designed an elective onboarding course, especially for this group of students, to ensure they feel welcome to participate and become computer scientists without being shied away from stereotypes or assumptions.

2. Pedagogical framework

Along with evaluating the kickstart experience for students, we also want to reflect on the eight design principles we used to develop this course over the past several years. The design of the course and the evaluation is based on two key concepts, which are defined in the following way:

Self-efficacy beliefs: Self-efficacy is typically defined as a person's belief in their ability to succeed in a specific situation or in a particular task (Bandura, 2000). It is one manifestation of how individuals come to perceive themselves from experiences and interactions with others and their capacity to have some degree of control over their environment. As such, self-efficacy is the foundation upon which student persistence is built. Students have to believe they can succeed in college. Otherwise, there is little reason to continue to invest in efforts to do so.

Sense of belonging: Believing in one's ability to complete a particular course of action, though essential to persistence, does not ensure persistence. It is also required that students come to see themselves as a member of a community or faculty, staff and other students who value their participation, that they matter and belong (Tinto, 1987).

Through the development of the kickstart course, the following eight design principles have guided how we teach. These design principles are derived from practice, discussion among instructed, and suggested loosely by theory, but mostly from practice and teacher discussion. Our starting point is the need to teach introductory computer science more as a computational thinking set of skills rather than a set of tools, paradigms, and programming languages.This practice grows on the tradition of the department and its approach to the educational challenges of teaching computer science (Sveinsdottir and Frøkjær, 1988). The aim is to create experiences for the students that stress the science of computations, rather of all types of data and data processes, and to move the focus away from computers as being central to this endeavor (Caeli and Yadav, 2020).

1. Self-directed learning process: Our first principle is to adopt a social constructivist approach. We, as teachers, should act as facilitators and only use limited portions of the course in lectures (below 20%). Instead, we should provide students with only the necessary information, tools and guidance to actively learn by working hands-on, as well as give them ample opportunity to get feedback on their work.

2. Faded worked examples: To allow for a self-directed learning process, we have designed a set of worksheets1 as a guiding tool for the students. The worksheets have been designed using faded-worked examples (Atkinson et al., 2003), where the student goes through four phases for every new programming concept (e.g., variables, conditionals, iteration):

- Typing in code verbatim using the new concept
- Editing the code to do certain things, to understand the new concept better
- Solving exercises using the concepts they just learned
- Doing a small creative project.

For every new concept, we start with as much structure as possible (typing in code) to support students in internalizing the syntax and how to use a text editor. We then fade the concept until students experience almost no structure (doing a project). Going through these steps, the students should also experience more and more freedom to experiment with the code and fewer and fewer strict requirements about how the code should function. This allows students to feel in control and obtain ownership over their code which is our following and third guiding principle.

3. Sense of ownership and self-efficacy: To support our second goal of developing student self-confidence in their ability to learn to program; we should design the material in a way where students obtain a sense of ownership of the programs they are writing as well as strengthen their self-efficacy

(Bandura, 2000). This requires us to ensure students have intimate knowledge about their entire programs and feel free to impact their program as they like. As mentioned in the previous section, we allow for such freedom and encourage students to modify the programs as they like and do their creative projects. To ensure students have ownership of their entire programs, we try to avoid handing out code but let them type in from worksheets or as part of "code-a-long" sessions, where everyone constructs a similar program together.

4. Pair programming and code-reviews: Our fourth design principle is to challenge students' assumptions and expand their perspectives on how computer scientists work by introducing them to pair programming and code-reviewing. Pair programming has been a valuable part of our course, as it also fosters collaborative learning and strengthens social bonds, as well as creates opportunities for students to talk about their programs with one another and start learning the vocabulary around programming ("Perhaps we should declare a new variable"). It should be noted that the discussions do not happen out of the blue, and the teaching assistants should help guide and foster dialogue among students. To further support students, in addition to what they can learn from the feedback they get from teaching assistants, we introduce daily peer-review sessions where two pair programming groups meet and exchange code. The peer reviews are structured through a checklist that explains what the reviewers should look for in the other group's code. The daily session allows the student to get their code in better shape, making it easier for them to continue working on their projects the following day.

5. Tangible Data: To support our goal of broadening perspectives on computer science, we want to show students that what a computer scientist does is not limited to what happens on screens. We want them to experience that they, as future computer scientists, can impact the physical world and experience data as the bridge between the physical and digital worlds.

6. Generate relevance: To support further the goal of broadening perspectives on computer science. It is also essential to show students that computer scientists can help solve some of today's societal issues. In our course, we have decided to use the central portion of the second week on a design thinking-based hackathon (Plattner, 2013), where students brainstorm, plan, design, program, and prototype a small Internet-of-things product. The week ends with a small exhibition. We have used sustainability as a theme for the hackathon, as there are many opportunities for students to develop ideas around environmental (shorten showers) and social sustainability (reduce stress). We could just as well have used any other theme.

7. No homework: The first steps in learning programming can and often will be frustrating, with many issues where students get stuck and need help. Working on your own is not ideal in those circumstances, as you need access to help when problems arise and you hit a roadblock. A common solution is automated feedback on everything from code style to correctness. However, these automated feedback tools only work for code, where there are one or few correct ways to solve a problem. When projects are open-ended and perhaps even include physical hardware such as sensors, automated correction falls short. Instead, we have scheduled enough time during the day for students to work on their projects and hired more teaching assistants per student than we traditionally do.

8. Worksheet on paper: Ten years ago, it might not be controversial to print teaching material for students. Still, today we are accustomed to digitally delivering our worksheets, assignments and other course material. However, for an introductory course in programming, where students' cognitive abilities are stretched to the limit, we need to be aware of all tasks we intend students to perform. Delivering worksheets digitally introduces extraneous cognitive load (Sveinsdottir and Frøkjær, 1988), as students working on small laptop screens are required to switch between the text editor and PDF viewer. For our course, we have thus decided to hand out all worksheets on paper to ensure students can continue to have their code on the screen while also looking at the worksheets.

3. Learning environment

The course happens in the 2 weeks leading up to the start of studies for new university students. The course ran for 2 weeks (9:00–15:00 every day), where the first week was composed of mainly hands-on work using Python mode in the Processing programming environment. The second week builds on their Python experience from the first week but moves on to programming Internet-of-things devices using MicroPython on ESP32 micro-controllers. The second week ended with a design thinking-based hackathon where teams of students developed interest-driven projects for the last two days.

3.1. Research/Evaluation questions

To investigate the kickstart course and the design principles, we formulated the following questions:

1. How do the students perceive the kickstart course?
2. How do the students experience their programming skills, and do they develop confidence in their ability to learn to program (self-efficacy)?
3. How do the students experience their start of studies (sense of belonging)?

3.1.1. Scope of project

The evaluation of the kickstart course was conducted as a development project to gain knowledge about the kickstart course, as seen from the student's perspective. Therefore, the

TABLE 1 Design principles.

	Principles	Motivation
1	Self-directed learning process	Active learning
2	Faded work examples	Self-directed learning
3	Sense of ownership and self-efficacy	Students develop confidence by creating their own projects from scratch
4	Pair Programming and code reviews	Dialogue and reflection on programming
5	Tangible data	Exploring modalities of programming
6	Generate relevance	Socially driven computing
7	No homework	Dev-camp and time to work
8	Worksheets on paper	Encouraging students to type in rather than copy/pasting

evaluation was carried out with a developmental perspective and with the overall purpose of gaining knowledge that could be used to develop further the course in close dialogue with both the responsibility and the department. Additionally, the evaluation was not conducted with a direct research purpose but rather in line with the aims of a Curriculum, Instruction, and Pedagogy (CIP) to inform the practice of introductory computer science courses.

Two of the authors have been responsible for the course's design and teaching. At the same time, the third author conducted the evaluation. It is essential to distinguish between the different roles and emphasize that the teachers have not been part of the course evaluation and that the evaluator was not part of designing the course. The evaluator has observed the teaching and more importantly, the students during the course. Additionally, they were solely responsible for recruiting the students from the kickstart course and the overall student body. The evaluator, the department, and the course responsible discussed the design and the evaluation planning. This resulted in the involved stakeholders identifying the focus and the purpose of the evaluation, formulation of evaluation/ research questions and the practical planning of the evaluation.

3.2. The kickstart course

The 2021 course had 117 students, two teachers, and six teaching assistants. The day starts with an inspiring mini talk about the diverse aspects of computer science and then the main topic of the day with the short lecture described by design principles 1, 5, and 6 (see Table 1). Students work in pairs in small teams of less than 20, guided by a teacher assistant using the worksheets (Table 1 principles 1, 2, 3, 4, 5, 6, and 8).

The goal of the kickstart course is not to educate proficient programmers but to make the students feel comfortable and less worried about starting their studies. After the course, students should have:

1. Reduced their knowledge gap in terms of programming experience when compared to students who start with prior experience
2. Developed self-confidence in their ability to learn to program
3. Has established a sense of belonging (Tinto, 1987) within the group of students and a member of the wider category of computer scientists to be
4. Broadened their perspectives on what computer science and programming are, what types of problems computer scientists solve, and how computer scientists work and collaborate.

3.3. Project design and data collection

The evaluation was designed as a qualitative study, consisting of group interviews and individual interviews with three groups of students: 1) Students with no/ limited programming experience who had attended the course, 2) Students with no/ limited programming experience who had not attended the course, and 3) Students with a lot of programming experience who had not attended the course. Qualitative classroom observations of the teaching and the student's participation in the kickstart course were conducted in August 2021. The data from the classroom observations have not been analyzed but have provided insights for the qualitative study and the interviews.

Based on the research/ evaluation questions, the group interviews focused on the students' experience of 1) their start of studies, 2) their programming skills, and 3) the kickstart course. The group interviews were conducted in September 2021, just after the start of the studies. Three rounds of group interviews were conducted in September, one round for each group of students. The students in the group interviews were recruited voluntarily by our physical presence in exercise classes on campus just before the interviews and *via* e-mail invitations.

The following number of students were recruited for each group of students:

- Nine students with no/ limited programming experience who had attended the course
- Five students with no/ limited programming experience who had not attended the course
- Five students with a lot of programming experience who had not attended the course.

In the group interviews, the students were divided into smaller groups of 2–3. The group interviews combined individual reflections on post-it notes, group discussions based on personal reflections and individual responses to an online survey. Each round of group interviews was facilitated by an interviewer so that the smaller groups answered and discussed the same questions simultaneously. Each round of group

FIGURE 1
Thematic analysis workshop notes in Danish, photograph by authors.

interviews lasted for approximately an hour. In each round of group interviews, the discussions in each smaller group were recorded. Each audio recording has been listened through, and notes have been written. Essential quotes are written in total length. A thematic analysis of the post-it notes from the individual reflections and the notes and the quotations from the group discussions have been thematically analyzed based on the research/ evaluation questions (Braun and Clarke, 2006).

In December 2021, individual interviews were conducted with seven students. Based on the research/ evaluation questions and the group interviews, the individual interviews consisted of questions on the students' experience of how things were going academically and socially. The interviews also focused on the student's experience of their programming skills and their perspectives on the kickstart course. All 19 students' from the group interviews were invited *via* e-mail to participate in an individual interview. 7 of these 19 students accepted the invitation. The following number of students were recruited for the individual interviews:

- Three students with no/ limited programming experience who had attended the course
- Three students with no/ limited programming experience who had not attended the course
- One student with a lot of programming experience who had not attended the course.

3.4. Data process

The individual interviews were conducted virtually in December 2021 and lasted 30–60 min. The individual interviews were recorded. The recordings were listened through, and notes were written. Essential quotes are written in total length. A thematic analysis of the notes and quotations from the group and individual interviews has been conducted (Braun and Clarke, 2006). The thematic analysis can be described as a theoretical thematic analysis driven by the research/evaluation questions rather than an inductive approach. The thematic analysis was conducted in two rounds, the first round with the primary purpose of developing an interview guide for the individual interviews and the second round focusing on analyzing all collected data (both group interviews and individual interviews). In the first round of the analysis, initial reading and analysis of the individual answers to the survey and the post-its from the group interviews were conducted. This initial analysis was driven by the research/evaluation questions, focusing on the perspectives of each group of students and led to the development of an interview guide for the individual interviews.

In the second round of analysis, all recordings of the group interviews and the individual interviews were listened through, notes were taken, and essential quotations were written at full length. The quotes from the students have been translated and

edited for readability. This second round of analysis was also driven by the research/evaluation questions and focused on the perspectives of each group of students. Figure 1 illustrates this process.

The thematic analysis can be described as a recursive process moving back and forth between the collected data, the codes, and the data extracts, keeping in mind that the evaluation was a development project rather than solely research. Both the first and second rounds of analysis were a way of familiarizing with the data. In the second round of analysis, notes from the interviews and the post-its were read through. Initial codes from the data were generated, and data extracts were organized into meaningful groups. The different codes have then been refined and sorted into initial themes, and the themes have been refined through a re-reading of codes and extracts from the interviews. A scheme for the thematic analysis was developed to analyse the data systematically. The agreement of the codes was done through discussions between the authors and the head of studies for the department. Keeping in mind that the project was focused more on organizational evaluation rather than for research.

4. Results to date

We used several interventions to investigate the three evaluation question to understand how students experienced the course, their programming skills, confidence, and the start of their university studies. A course evaluation survey was conducted, and 61 students participated. Additional, several weeks into the beginning of the semester, students were recruited for qualitative study consisting of group and individual interviews of the different groups of students. The following subsections present the results of these interventions.

4.1. Course evaluation survey

We conducted a course evaluation survey on the last day of class. The survey had five demographic questions and nine free text questions that provided opportunities to express how they experienced the course, aspects of the course they would change, the difficulty of the course, and the style of teaching and level of feedback.

4.1.1. Demographics

What we see in the demographics is the split between; we have three bachelor programs for computer science, machine learning, computer science and economics, and general computer science. From the end-of-term evaluation survey, 61 of the 117 students participated. Most kickstart students came from the broader computer science track (35), 19 from computer science and economics, and seven from machine learning. The gender split was typical, with 43 men, 19 women, and a single non-binary student. The student age also followed the normal distribution for university, with half (30) of the students being 21–23 year students from the 18 to 20-year-old bracket. Additionally, some outliners from 24 to 29 (9) and three students between the ages of 30 and 39. Most students (41) come from gymnasium, the broad secondary education programme at a high level, which qualifies students to go to university and other forms of higher education. Most importantly, 41 students have no programming experience for the kickstart course. These demographics are similar to the university's student body in computer science.

4.1.2. Open questions

For the paper, we will focus on the first open question, "How did you experience the course? Describe the course as you explain it to a fellow student who has not taken it." The students highlighted the kickstart program's goals, especially self-confidence and self-belonging. However, we wanted to get a more in-depth response and comparison between different levels of students and students that did not attend the kickstart course, which was one of our main motivations for conducting the evaluation. One of the responses highlights the teacher's aim for the course as follows:

> "The course provides fast and educational insight into programming. The first few days are enormously progressive, with good and well-prepared worksheets that guide one through the 'curriculum'. It worked well as an anxiety-reducing tool about one's expectations to start in computer science without experience. You get a basic understanding of the subject, that it is problem-solving through the instructions you write, and you learn a little about the general language. It is such a de-alienating course. It's a cool offer for someone unsure of what exactly to study, reassuring." (kickstart participant)

Another student summarized their feelings about the course highlighting the more social aspects that the kickstart aims to create:

> "It was also a casual course, so you did not feel overwhelmed by new information and also had time to socialize a bit with the others on your team. all in all, a super good course, which makes me feel much more ready to start studying, both purely academically about having learned programming 101 but also with a good focus on the social."(kickstart participant)

Additionally, the students reported on how the course provided opportunities for getting to know people by working together to learn about programming.

> "In addition, it is a course where during the week you get to know someone who is in the same place and you, therefore, work together to get a lot of learning." (kickstart participant)

4.2. Qualitative study-group interviews, a few weeks after the start of studies'

Students who have participated in the kickstart course experience that the course has made their start of studies easier socially and academically. The kickstart students have general reflections about the course in retrospect to the beginning of the semester and how they felt about the first few weeks of teaching, especially in programming:

> "I was happy with it [the kickstart course], at least. During the first week, it was easier; now is the second week, so the work is okay, and I feel that I am on a level with everyone else" (kickstart, no/limited experience)

At the same time, the kickstart course has strengthened the participants' confidence in their ability to learn how to programme. A student had the experience of being enabled through their participation in the course:

> Before I started studying, I was afraid I could not follow the teaching and could not think code-oriented. In addition, I was also a little scared that I would use a long time to find a group of friends I could be comfortable with. Both ideas have fortunately been something that was just in my head." (kickstart, no / limited experience)

Socially, participating in the kickstart course has also made the start of studies easier:

> ". . . It was nice [. . .] that I knew someone in advance. That made the study start (. . .) a little less awkward when meeting a lot of new people at the same time. That I think, was super fine." (kickstart, no / limited experience)

However, all students, both those who have attended the course and those who have not, experience that students, who have participated in the course, have the advantage of knowing someone in advance during the first weeks of their studies. A student who had not participated in the course stated:

> ". . . Half of the class already know each other. . . It creates another dynamic. Usually, no one knows anyone in advance at the start of their studies. It is easier to get to know each other in the first weeks." (non-kickstart, no / limited experience)

This causes a social gap between the students who have participated in the course and those who have not, at least in the first weeks of the study programme.

4.3. Individual interviews at the end of the first semester

At the end of the first semester, in early December, seven students were recruited, three from the kickstart, three non-kickstart with limited experience, and one non-kickstart with a lot of experience. Students who participated in the course continue to experience that the kickstart course made their start of studies easier academically and socially. Kickstart students in December felt that the course gave them a sense of belonging that helped them transition to university.

> "Starting studying in a new place, it takes some time to get to know the place, but maybe it was easier for me, because I already knew the place a bit." (kickstart, no/limited experience)

Additionally, the peer programming used in the kickstart course provided students with added skills to collaborate more effectively with other students, providing students with more tools to help them with their studies.

> "There is also some peer programming, which has been quite nice to do in group work. Because it's much easier to spot other people's mistakes than to spot your own, pair programming has been very nice in group work." (kickstart, no/limited experience)

Students also reflected on how the kickstart, the start of studies, and the different courses allowed them to move from other groups and meet new students.

> "In the team, it might have meant a bit of a team feeling that we knew each other beforehand. I don't know if it has made a difference, maybe. It certainly hasn't had the negative effect that you only stick with people you know hand. I don't think that's been the case. People are open whether they have been there or not." (kickstart, no/limited experience)

Both students who have attended the kickstart course and those who have taken the course experience no difference socially or academically in December.

5. Discussion

To answer our first question of how students perceive the kickstart course, the findings suggest that the kickstart course reduces the gap in programming experiences

and strengthens students' self-efficacy and sense of belonging. However, the learning experience initially creates a social gap between students who have not attended the course and those that participated at the beginning of the first semester. This gap is especially prevalent for students with little or no prior programming experience who did not attend the course. However, by the end of the semester, the students do not perceive any differences.

The second question is about how the students experience their programming skills and do they develop confidence in their ability to learn to program (self-efficacy). We see that the students gain confidence through the experience, and, to some extent, the design principles help guide the students by allowing them the experience of learning to program in a less pressured space, a voluntary course giving them the freedom to experiment more. We believe that our pedagogical framework allows students to have self-efficacy as defined by Bandura (2000).

For how students perceive the start of their studies (question 3), we see a critical point for discussion since the gap between students with no or little programming experience that have or have not attended the kickstart creates a new dilemma in terms of the purpose and the aims of the kickstart. Recognizing and addressing this dilemma of feeling left behind by the students that did not participate in the kickstart further highlights the complexity of designing and running these preparatory courses.

Statistics from the university administration have illustrated that the kickstart course does not significantly change the drop-out rate. But, the department sees the benefits of providing this course to students with limited programming experiences by giving them a step toward confidence and a sense of belonging. We are looking at further expanding the kickstart course's principles to reach more students and for ongoing outreach programs.

5.1. Lessons learnt

In retrospect, over the last several years of running this course, we believe universities need to provide more robust scaffolding to help students transition to higher education. Opportunities for kickstart or development camps can provide this opportunity if designed in a way that provides students with the self-confidence to tackle complex problems. In Table 1, we summarize the design principles and the motivations for using these principles. These principles have been developed through practice to support the department's vision for computer science for research and education (Caeli and Yadav, 2020).

Additionally, when further designing the two first courses all the computer science students take, we have discussed how to integrate aspects of these principles into the introductory courses for the bachelor programs in computer science and other introductory programming courses. There has been close dialogue between the course designers, teachers, and the evaluator with the head of studies and the other teachers in the department. Additionally, the kickstart materials and practice have been used in other non-computer science introductory programming courses to help these students learn to program and computationally think. These problems are common to computer science education, and educators and researchers approach them differently. What is essential is the knowledge sharing that both supports innovative education and research (Schulte and Knobelsdorf, 2007; Robins, 2019; Von Hausswolff, 2022).

The research reported in this paper has limitations around the methodological approach since we investigated the kickstart course from the pragmatic position of evaluating benefits for the department. Additionally, the university and the department have a particular approach to pedagogy that the course aimed to introduce to the students, with the notion of lecturers, teaching assistants, and worksheets.

5.2. Contributions

To summarize the paper's contributions, first, we see the benefits of providing kickstart courses that use development and boot camp approaches to give confidence and self-efficacy to new students. Mainly as a transition experience for incoming university students. Secondly, the practice-based design principles encourage strategies for computational thinking that include problem-solving, active and self-directed learning, and discursive and reflective processes like pair programming combined with real-world problems to encourage the students. These approaches can help students feel a sense of belonging to their studies and relevance to society, creating more capable learners. However, when designing a voluntary course recognizing the students that cannot attend may be adversely affected in the beginning needs to be considered.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Ethics statement

Ethical review and approval was not required for the study on human participants in accordance with the local legislation and institutional requirements. The patients/participants provided their written informed consent to participate in this study.

Author contributions

DS contributed by bringing together the course design and the evaluation. MD designed the course and developed the principles. DE conducted the evaluation.All authors contributed to the article and approved the submitted version.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Atkinson, R. K., Renkl, A., and Merrill, M. M. (2003). Transitioning from studying examples to solving problems: effects of self-explanation prompts and fading worked-out steps. *J. Educ. Psychol.* 95, 774. doi: 10.1037/0022-0663.95.4.774

Bandura, A. (2000). "Self-efficacy: the foundation of agency," in *Control of Human Behavior, Mental Processes, and Consciousness: Essays in Honor of the 60th Birthday of August Flammer, Vol. 16* (Mahwah, NJ: Erlbaum), 17–33.

Bennedsen, J., and Caspersen, M. E. (2007). Assessing process and product. *Innovat. Teach. Learn. Inf. Comput. Sci.* 6, 183–202. doi: 10.11120/ital.2007.06040183

Braun, V., and Clarke, V. (2006). Using thematic analysis in psychology. *Qual. Res. Psychol.* 3, 77–101. doi: 10.1191/1478088706qp063oa

Caeli, E. N., and Yadav, A. (2020). Unplugged approaches to computational thinking: a historical perspective. *TechTrends* 64, 29–36. doi: 10.1007/s11528-019-00410-5

Kandemir, C. M., Kalelioğlu, F., and Gülbahar, Y. (2021). Pedagogy of teaching introductory text-based programming in terms of computational thinking concepts and practices. *Comput. Appl. Eng. Educ.* 29, 29–45. doi: 10.1002/cae.22374

Luxton-Reilly, A., Simon, A.lbluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., et al. (2018). "Introductory programming: a systematic literature review," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (ACM), 55–106. doi: 10.1145/3293881.3295779

Plattner, H. (2013). "An introduction to design thinking," in *Iinstitute of Design at Stanford* (Berkeley, CA: Apress), 1–15.

Robins, A. V. (2019). "12 novice programmers and introductory programming," in *The Cambridge Handbook of Computing Education Research*, eds S. Fincher, and A. Robins (Cambridge: Cambridge University Press), 327. doi: 10.1017/9781108654555

Schulte, C., and Knobelsdorf, M. (2007). "Attitudes towards computer science-computing experiences as a starting point and barrier to computer science," in *Proceedings of the Third International Workshop on Computing Education Research-ICER '07* (ACM Press), 27–38. doi: 10.1145/1288580.1288585

Sveinsdottir, E., and Frøkjær, E. (1988). Datalogy–The Copenhagen tradition of computer science. *BIT Numer. Math.* 28, 450–472. doi: 10.1007/BF01941128

Tinto, V. (1987). *Leaving College: Rethinking the Causes and Cures of Student Attrition*. Chicago, IL: University of Chicago Press.

Von Hausswolff, K. (2022). Practical thinking while learning to program-novices' experiences and hands-on encounters. *Comput. Sci. Educ.* 32, 128–152. doi: 10.1080/08993408.2021.1953295