Routledge
Taylor & Francis Group

Pedagogies: An International Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/hped20

Making coding meaningful: university students' perceptions of bootcamp pedagogies

Georgie Tarling, Ana Melro, Judith Kleine Staarman & Taro Fujita

To cite this article: Georgie Tarling, Ana Melro, Judith Kleine Staarman & Taro Fujita (2022): Making coding meaningful: university students' perceptions of bootcamp pedagogies, Pedagogies: An International Journal, DOI: 10.1080/1554480X.2022.2077338

To link to this article: https://doi.org/10.1080/1554480X.2022.2077338

Published online: 02 Jun 2022.

PEDAGOGIES: AN INTERNATIONAL JOURNAL
https://doi.org/10.1080/1554480X.2022.2077338

Routledge
Taylor & Francis Group

Making coding meaningful: university students' perceptions of bootcamp pedagogies

Georgie Tarling[a], Ana Melro[b], Judith Kleine Staarman[a] and Taro Fujita[a]

[a]Graduate School of Education, University of Exeter, Exeter, UK; [b]Departamento de Ciências da Comunicação e Tecnologias da Informação Universidade da Maia, ISMAI, Portugal

ABSTRACT

Coding bootcamps targeting diverse learners are increasingly popular. However, little research has focused on the student experience of these courses: what pedagogic practices make learning coding meaningful for them and why. In a previous paper, we proposed a conceptual framework outlining three dimensions of learning opportunities in relation to coding: functional, collaborative, and critical & creative. The aim of the current paper is to characterise the pedagogic practices and learning outcomes highlighted by university coding bootcamp students as beneficial and then explore how they might map on to the dimensions in this framework. This study uses a mixed method design with quantitative and qualitative data from a sample of 36 students attending a Summer School in Coding, Data Analytics and Machine Learning. We use thematic analysis to describe how three main pedagogic approaches – guided hands-on, peer learning and contextualisation – were experienced by learners. The study suggests that mapping pedagogic practices to the dimensions of the framework can be of value to those designing courses to suit the diverse interests of students seeking an introduction to coding.

ARTICLE HISTORY
Received 20 May 2021
Accepted 29 October 2021

KEYWORDS
Coding; pedagogic practices; bootcamps; learning opportunities

Introduction

Computation and data play a central role in the lives of today's young people (Barr & Stephenson, 2011). Artificial intelligence and coded processes are transforming employment, helping to solve pressing problems and spurring innovation across a range of new interdisciplinary fields from computational biology to computational economics (Czerkawski, 2015; Grover & Pea, 2017). The use of algorithms in data-driven decision-making also demands critical interrogation (Baker-doyle, 2018) and awareness-raising around issues of diversity and bias (Dempster et al., 2020). To be prepared for future work as well as to participate in society, it is therefore urgent that today's university students have a basic understanding of how computational processes work and what they can be used for (McCord & Jeldes, 2019). Since a majority of undergraduates have not benefitted from any computing education in their schooling, there is a need for rapid upskilling (Czerkawski, 2015; Lockwood & Mooney, 2018; Peteranetz et al., 2019; Pollock

CONTACT Georgie Tarling G.Tarling3@exeter.ac.uk Graduate School of Education, University of Exeter, Exeter, UK

et al., 2019). Demand for computing courses from students in disciplines outside of Computer Science (CS) is therefore growing (Camp et al., 2017; Dawson et al., 2018; Sax et al., 2017).

One response has been to create courses or modules tailored to specific disciplines (Pollock et al., 2019; Soh et al., 2009). This has been shown to make content motivating to students (Kafura et al., 2015; Soh et al., 2009), and to better equip them to address problems at the intersections of disciplines (Barr, 2012). This approach also promotes accessibility and inclusion, showing students that there are multiple ways to be a computer scientist (Falkner & Sheard, 2019; Margolis et al., 1999). Some universities report improvements in enrolment, gender diversity and engagement following overhaul of the curriculum to be more interdisciplinary (Barr, 2016). Another approach for making coding relevant to learners is the bootcamp model (Burke & Bailey, 2020), which focuses on developing timely, entry level, technical skills favoured in the marketplace (Waguespack et al., 2018). Bootcamps can be appealing to non-CS learners for a number of reasons, primarily because they are short and focused on employability, but also because they can be less intimidating than taking CS courses (Lyon & Green, 2020).

However, few studies foreground the views of students themselves on what makes learning coding relevant to their lives and careers (Dawson et al., 2018; Dempster et al., 2020; Santana & Bittencourt, 2018). This is especially the case with regard to those just starting out and still unclear about what learning coding has to offer (Lehman et al., 2018; Sax et al., 2017). In a previous study, we explored university students' reasons for taking a coding bootcamp and found that they perceived coding as a transferable skill with potential value for employability, for participating in interdisciplinary conversations, and for being informed about the impact of coding in society (Tarling et al., 2020). Bringing this data together with literature on CT, we created a conceptual framework that shows how introductory coding courses can offer opportunities for learning in three dimensions – functional, collaborative, and critical & creative (Tarling et al., 2020).

In the current paper, our aim is to build on this previous work and characterise specific pedagogic practices that support the learning of and about coding for non-CS students. We draw on data from a higher education bootcamp in coding, data analytics and machine learning to answer the following research question:

(RQ): *What pedagogic practices do non-CS students identify as meaningful when learning coding and why?*

We then map these findings onto our framework to enhance its relevance and usability.

Our study has value for higher education practitioners and designers, particularly those seeking to increase undergraduates' awareness of the affordances of coding by hosting short, introductory courses (Seibel & Veilleux, 2011).

Literature review

Pedagogic approaches for engaging non-CS learners in coding

Several studies focus on practices to make computing concepts more accessible to non-CS students. Some have limited the amount of coding in the curriculum (Peteranetz et al., 2019) and instead teach explicitly how to abstract tasks and decompose problems

(Czerkawski, 2015; Luxton-Reilly et al., 2018). Others use repeated exposure to underlying computational ideas in different forms (Kafura et al., 2015). Studies report that revisiting the same concepts with increasing level of difficulty helps to reduce cognitive load (Santana & Bittencourt, 2018) and students appreciate having multiple opportunities to interact with course concepts and ask for help over time (Dawson et al., 2018). Often underpinning these approaches is the concept of computational thinking (CT), a way of understanding the processes behind operations in order to solve problems with deeper knowledge (Grover & Pea, 2017). A key factor here is the extent to which students are supported in their metacognitive or reflective activities (Sanders et al., 2017), it being argued that students ought to be thinking-doing and not just doing (Lye & Koh, 2014). Encouraging students to review their problem-solving thought process is also seen as an opportunity to build communication, collaboration and feedback skills (Falkner & Sheard, 2019). However, multiple studies have shown that students do not necessarily develop these skills without guidance (Burke et al., 2016; Carter, 2011) and that they can underestimate the value of these skills in computing contexts (Cohen et al., 2017; Seibel & Veilleux, 2011).

Another approach, reported widely in studies of bootcamps, is the use of tasks which mimic real-life projects (Burke et al., 2018; Lyon & Green, 2020), and which foster a culture of errors that allows for immediate correction and learning (Lyon & Green, 2020). Crucial to the success of this approach is having supportive staff to continuously monitor student progress (Amaro et al., 2020; Stefan et al., 2015). A similar method combining small group exercises and just-in-time teaching has been adopted in university courses for non-CS learners (Dawson et al., 2018; Stefan et al., 2015), with instructors on hand to clarify doubts and help overcome barriers (Buitrago Flórez et al., 2017; Kafura et al., 2015; Seibel & Veilleux, 2011). Studies also report on the success of peer mentors or Teaching Assistants (TAs), whose closeness to the student experience helps build rapport, providing encouragement and explaining things in ways students find easier to understand (Pollock et al., 2019; Pon-Barry et al., 2017; Vihavainen et al., 2014). These courses often centre on student-defined projects using real-world data to increase motivation (Dawson et al., 2018; Kafura et al., 2015). Working on projects allows students to apply skills to their personal interests and to tackle problems more generally (Dawson et al., 2018).

Also reported in the computing education literature is the use of collaborative learning, which can take different forms. For example, a review of the literature on pair programming with university students shows they typically report fewer programming errors, learn from each other and are more confident than those who program alone (Hanks et al., 2011). Another method is the 'cohort model", in which students solve problems individually within a cohort designed to act as a supportive sub-community (Chowdhury et al., 2018). This allows time for the long-term building of relationships, active peer learning with explicit guidance on interaction, online collaboration tools and a "contract" of responsibilities (Chowdhury et al., 2018). Overall, it is argued that students find it easier to ask a peer to explain a problem because they are at the same level of learning, and that discussing problems with students from other disciplines allows them to see how differently students perceive and explain a problem (Chowdhury et al., 2018; Kafai & Burke, 2014). More broadly, collaborative learning can help reduce learners' level of anxiety about coding, improve understanding and thus create a positive atmosphere for learning

(Chowdhury et al., 2018; Dawson et al., 2018; Seibel & Veilleux, 2011). However, in order for collaborative practices to be effective, students need to be explicitly taught not only how to program but also how to collaborate (Burke et al., 2016).

Learning opportunities through coding: a conceptual framework

Some studies on pedagogic approaches tailored to making programming accessible for non-CS learners focus also on promoting the development of skills such as teamwork, communication, interdisciplinary understanding for problem solving, and the ability to listen and negotiate different perspectives or build consensus (Chowdhury et al., 2018; Czerkawski, 2015; Luxton-Reilly et al., 2018; Lyon & Green, 2020; Soh et al., 2009). However, none of these studies present a framework to capture these wider learning opportunities within coding courses and identify the pedagogic practices to support them. In a previous study using student voice data, we argued that introductory coding courses should not just be about teaching students to code or preparing them to become computer scientists. Rather, they should be about giving students a taster of what coding might be for them, how they might use coding in their own work, and how they might critically and actively engage in coding contexts. Bringing this data together with literature on CT, we created a conceptual framework that shows how introductory coding courses can offer opportunities for learning in three dimensions – functional, collaborative, and critical & creative (Tarling et al., 2020). As shown in Table 1, in the functional dimension, learning coding is centred on the individual development of skills related to programming, such as decomposition, abstraction, debugging, logical thinking or problem-solving. The second dimension frames the learning of coding as a collaborative process where students develop communication and teamwork skills to engage with peers and in coding communities. The third dimension focuses on the development of critical and creative skills through coding by fostering awareness of the impact of coding in society and promoting innovative problem-solving.

In this paper, we explore the pedagogic practices perceived as meaningful by non-CS students for learning with and about coding. We then map these practices onto our framework to enhance its operational value and relevance to learning designers and teachers.

Methodology

Study background and participants

The Institute of Coding is a UK initiative to support students from a wide range of disciplines to upskill in computing (Davenport et al., 2019; Dempster et al., 2020). For the last three years a team of researchers have been iteratively studying a Summer School in coding (Block 1 or B1), social data analytics (Block 2 or B2) and machine learning (Block 3 or B3). The aim of the bootcamp was firstly to provide a foundation in programming (Python and R), secondly to signpost why this might be useful for students' lives and for different disciplinary or workplace contexts, and thirdly to introduce some opportunities for collaboration and critical discussion. Each day would alternate between short lectures and workshops and was supported by TAs and guest speakers from industry and

Table 1. Multidimensional framework of learning opportunities through coding based on previous empirical data and CT literature review.

CT dimension	Learning opportunities	Description
Functional *Individual/ self-centred* (Coding)	Problem-solving Abstraction Decomposition Debugging Algorithmic reasoning Mathematical thinking	Coding is seen as fundamental for employability in today's workplace. Learning is placed in getting an introduction to new vocabulary and core elements of CT – abstraction, decomposition, debugging, algorithmic thinking, mathematical reasoning and problem-solving.
Collaborative *Community-centred* (Coding with others)	Communication Collaboration Teamwork Interdisciplinary Participation	Coding is seen as relevant to engage in conversations around technology developments and participate in communities and public discussions about AI and data science. Learning is placed in developing collaboration and communication skills through and about coding.
Critical & creative *Society-centred* (Coding for others)	Creativity Critical thinking	Coding is seen as important to make informed decisions as citizens, to have a critical understanding about the impact of coding in society. Learning is placed in helping students be critically aware and finding informed and creative solutions to solve problems.

academia. In addition, each block is built towards a group project and presentation on the final day. This paper reports on the 2019 iteration of the course. Although the framework was not fully developed at that point, the design of the 2019 iteration was informed by the findings of the previous year. The learning objectives displayed on the course website[1] primarily referenced functional learning outcomes, for example, to be able to "read and write files". However, the intention to foster collaborative work was built into the design of the course through a consistent focus on pair work and each week's group project. Through the latter, students were tasked with creatively generating their own application. The projects and guest lectures were also intended to stimulate critical discussion, both about career trajectories and about the implications of coding applications.

There were 38 students enrolled in the course, of whom 36 consented to participate in the research. Students were given the choice to attend one, two or three blocks of the course. As described in Table 2, students were relatively equally distributed across all blocks, with B2 being the most popular. About two in five students attended all three blocks (39%; N= 36).

Our sample (N = 36) is mostly composed of 25-year-old (72%) female (57%) undergraduates (72%) from social sciences (67%). The range of disciplines is diverse, with Politics and Economics being the most common. Overall, in a scale from 1 to 4, students reported being "very interested" (N = 36; M = 3.7; SD = 0.9) in data science or programming

Table 2. Number of students per block in the 2019 institute of coding summer school.

Students per block (N = 36)	*n*	%
Block 1 (B1) Coding Boot Camp	24	67%
Block 2 (B2) Social Data Analytics	26	72%
Block 3 (B3) Artificial Intelligence and Machine Learning	21	58%

before taking the course. They described a range of roles in which they could apply programming or data analytics, both within their current field or in their further careers. General curiosity in becoming more skilled also played an important role in leading students to take the course.

Research tools and methods

In this paper, we primarily use qualitative data from both semi-structured interviews conducted with students (*N*= 14) and open-ended questions from a questionnaire (*N*= 36), both collected at the end of each week of the course. The student interviews focused on their reasons for taking the course, their experience of the course itself and their perceptions of what they had learnt. The questionnaire was centred on student perceptions of the teaching and content in each block (B1, B2 and B3). Using a mixed method approach with a triangulation design (Creswell & Clark, 2006), we first performed inductive content analysis of each dataset separately, using NVivo, and then combined the resultant codebooks (*N* = 36), allowing for joint interpretation of the qualitative data. Additionally, we present the results of the descriptive quantitative analysis of the questionnaire[2] to contextualise some of the qualitative findings. All students were identified using an ID number, allowing to trace them anonymously across the different research tools: interviews (INT) and questionnaires (QUEST). The student quotes used here show that ID number followed by student gender, age, and field of study.

The combined content analysis from questionnaires and interviews used broad categories to describe the main pedagogic approaches employed in the course (*guided hands-on learning, peer learning*, and *contextualisation*). Within these, we characterise what students said about the pedagogic practices they experienced, as well as the learning opportunities developed in the course. Table 3 presents the three overarching approaches followed by the corresponding subcategories identified in the content analysis and brief description of the practices, presenting the number of cases or number of times they were referred by students.

In each section of the findings that follow, we explore student perspectives on meaningful learning opportunities with coding and the granular practices which supported them. We have aligned each of the main pedagogic approaches used in the course with one of the dimensions identified in our model – functional, collaborative, and critical & creative. However, we also highlight how practices support multiple learning dimensions.

Findings

Pedagogic practices and learning opportunities

Guided hands-on practices and functional learning opportunities

A consistent finding was that students identified hands-on practical work as a key practice supporting functional learning. However, although students liked having time to explore, they preferred this to be guided through well-defined tasks with clear expected outcomes (*structured workshops)*. It was important for students that the lecturers took time to explain the tasks, and particularly to ensure that the language used in any examples matched the tasks. When articulating what made practical work constructive, four

Table 3. Combined content analysis from student interviews and questionnaire.

Pedagogic approaches	Subcategories	Description of practices	References	Cases ($N = 36$)	%
Guided hands-on learning	Structured workshops	Well defined practical exercises and time to explore	46	23	63.9%
	Incremental explanation	Live coding, clear links between concepts and tasks, building on prior learning	92	26	72.2%
	Proactive feedback	Iterative cycles, recaps, just in time support from TAs and tutors	44	20	55.6%
Peer learning	Shared tasks	Encouraging collaborative problem solving and debugging through practical work	39	20	55.6%
	Social learning community	Fostering participation through bonding activities, signposting online communities, inclusive environment	42	22	61.1%
	Group work management	Explicit support for interaction, appropriate pairing and group formation	44	19	52.8%
Contextualisation	Real world resources	Use of real-world examples and professional advice, through guest lectures, datasets and tasks	33	17	47.2%
	Project working	Simulated project development for generating ideas, finding creative solutions and managing design and delivery of products	6	3	8.3%
	Applied discussion	Use of case studies as catalyst for critical debates	16	12	33.3%

students highlighted tasks that were designed to help them take "one step at a time" (ID31, INT, female, 23 years old, English) or "decompose the problem into really basic steps" (ID27, INT, female, 19 years old, Economics/Business). If students felt they were "just faced with a huge chunk of code, it was easy to lose sight and become frustrated" (ID2, QUEST, female, 24 years old, Politics/International Relations).

The helpfulness of *incremental explanation* was also observed in terms of the structure of the course more generally, with students talking of the need for content to "start at basics and build on" (ID7, QUEST, female, 20 years old, Sociology) and "make sense in the order we are doing it" (ID12, INT, male, 24 years old, Economics/Business). Over half of students (56%; *N*= 36) said they would have liked an introduction or "cheat sheet" of key concepts and vocabulary at the beginning of the block, as a resource for ongoing consultation: "just to know what word would correspond to what I wanted to do" (ID21, INT, female, 29 years old, Politics/International Relations). In spite of the difficulty revealed in certain moments, students articulated that explanations had in general been clear and understandable. As shown in Table 4, a factor analysis,[3] comprising 13 variables or statements from the questionnaire about the teaching in all blocks (N = 36), on a scale from 1 ("strongly disagree") to 2 ("disagree") to 3 ("neither agree nor disagree") to 4 ("agree") to 5 ("strongly agree"), identified two correlated types of perceived "teaching styles": (1) *explanatory* teaching (accounting for 51% of the variance) suggesting tutors were well prepared, and knowledgeable, answering questions completely and providing real-life examples; and (2) *engaging* teaching (accounting for 14.5% of the variance) with statements regarding how tutors created interest and enthusiasm, encouraged participation, drew on students' experience, encouraged self-learning and mixed workshops with lectures.

Table 4. Factor analysis of a closed question in the questionnaire about the teaching in all blocks in a scale from 1 to 5 (N = 36).

The tutors in this block	Factor 1 (51%) *Explanatory*	Factor 2 (14.5%) *Engaging*
Knew the subject area	0.81	
Used real-life examples	0.64	
Answered questions completely	0.75	
Provided clear explanations	0.77	
Were well-prepared	0.66	
Made me enthusiastic to learn		0.70
Created interest in the topic		0.63
Encouraged participation		0.72
Used an effective mix of lectures and workshop		0.58
Encouraged to learn more by myself after the course		0.74
Drew on my knowledge and experience		0.56
% variance explained	51.0	14.5
Cronbach's alpha	0.87	0.83

One practice that students highlighted as helping both explanation and engagement was the use of demonstration, with live coding in R Studio, walkthroughs in Python and use of Jupyter notebooks being described as "suit[ing] my style of learning" (ID10, INT, male, 21 years old, Politics/International Relations) or "the best way to learn" (ID14, INT, male, 19 years old, Economics/Business). Commenting and the frequent use of print statements were mentioned as helpful for making visible the alterations that each step of the code made: "it's like in maths at school, you have to show the workings out. You can't just have the answer, you have to show each step" (ID1, INT, female, 24 years old, History).

The final aspect of structured support highlighted by students was *proactive feedback*. The high ratio of teaching assistants (TAs)[4] to students meant that they never had to wait long for help: "I felt supported. They would come over pretty frequently to check in on us and ask whether we had any issues" (ID28, INT, female, 21 years old, Politics/International Relations). Students mentioned a range of ways in which TAs offered support including "explaining the tasks" (ID23, INT, male, 21 years old, Politics/International Relations), "helping to correct errors that I made" (ID15, QUEST, female, 21 years old, Biology), "making the content more understandable by explaining in different ways" (ID17, INT, female, 22 years old, Politics/International Relations) and simply reassuring 'even if I was sat there going "I can't do it, I give up", they'd come round and be like "It's fine, this is normal, do it this way" (ID8, INT, NA, 21 years old, Engineering, Mathematics and Physical Sciences). Students equally highlighted the importance of frequent checks on progress from the main tutors, with 'dedicated slots to look at the answers and a bit of time for everyone to go, "okay, I did not get that" or "I got so far" (ID12, INT, male, 24 years old, Economics/Business). Students also said they needed concepts and vocabulary to be recapped more than once to embed understanding: 'they would explain, but because there was so much jargon, some of it I would forget (ID10, INT, male, 21 years old, Politics/International Relations).

Broadly speaking, hands-on exploratory practices allowed students to learn a basic functional understanding of programming. One student summarised this by saying: "I have learnt how to at least write a string of code, I have learnt how to take information from a JSON file, put it into Python, extrapolate data to show what I want to show based

on the type of information that is in that file" (ID20, INT, male, 34 years old, Social Sciences). Particularly in B2, students identified concrete data management skills they had learnt, including importing datasets, extracting specific data, using packages and presenting data. Learning how to visualise data using ggplot (a data representation package in R) was mentioned six times as a key useful skill students had taken away from the course: "(. . .) doing visualisation was really good because in practice I can take a spreadsheet, find some interesting things in there, and then make it a finished product" (ID31, INT, female, 23 years old, English).

By the end of the course students had gained a foundation or overview of the basics. One student said she had "an idea of what is possible, like what are the options, what can you do with data" (ID31, INT, female, 23 years old, English). Several mentioned that the course had provided a roadmap and given them a clearer sense of purpose in terms of what they needed to learn: "I may not be able to do everything . . . but I understand how I can build on that" (ID20, INT, male, 34 years old, Social Sciences).

Overall, hands-on practices allowed students to overcome their nervousness about coding by realising that they "just need a sense of determination to keep learning and keep being persistent" (ID23, INT, male, 21 years old, Politics/International Relations). Although computational concepts were not taught explicitly, the majority of students (10 out of 14) described their learning in terms of developing a way of thinking, particularly in terms of problem solving and logic. Some struggled with this because "it didn't come naturally as a way of thinking, but [they] could see that the process involved breaking things down into steps" (ID28, INT, female, 21 years old, Politics/International Relations). In addition, debugging or "analysing your code back for errors" (ID12, INT, male, 24 years old, Economics/Business) was a crucial skill identified.

Peer learning and collaborative learning opportunities

Shared tasks were valued by students with results from the questionnaire across all blocks (N = 56) revealing that they found working in groups "very helpful" with an overall mean of M = 3.8 (SD = 1.1) on a 5-point scale. Students gave a high rating across all blocks on statements such as "discussed things together" (M = 4.4; SD = 0.7), "listened to each other" (M = 4.3; SD = 0.8), "helped each other solve problems" (M = 4.2; SD = 0.7) and "discussed alternative solutions to the problem" (M = 4.2; SD = 0.8). Equally, in the combined qualitative data, the majority of students (64%, N = 36) enjoyed learning with their peers and mentioned different ways in which this helped develop their understanding. In terms of debugging, for example, students found it good to work with others because "having another pair of eyes made it easier to spot tiny errors" (ID20, INT, male, 34 years old, Social Sciences). Students liked that partners had different ways of thinking, for example, stating that "learning from different perspectives helped me tackle problems" (ID15, QUEST, female, 21 years old, Biology). Another commented that he sometimes "relied on other students for explanations of what was going in the code" (ID37, INT, male, 34 years old, Education). In some cases, working in pairs offered a safe space for testing ideas, with one student appreciating that "as someone not very confident with my ability to code it was great to get feedback from the others that I was thinking on the right lines, and we could work it through together" (ID28, INT, female, 21 years old, Politics/International Relations). As a result of working collaboratively in this context, one student concluded "I could confidently work in another group outside of this doing coding" (ID6,

INT, male, 22 years old, Politics/International Relations). Communication was mentioned as a key skill in coding, for instance, one student had realised that she was now able to translate from "those who speak the language of computers and those who speak human English" (ID1, INT, female, 24 years old, History).

In general, students were positive about the ways in which social bonding and a sense of community (*social learning community*) were fostered, for example, saying that an inclusive environment had been key to fostering open dialogue and commenting on the importance of icebreakers and social events as a way of fostering greater participation in class: "it has felt very much like we are all in this together, we are all learning it together" (ID17, INT, female, 22 years old, Politics/International Relations). The collaborative learning environment was also reflected in the ways students engaged in online coding communities, for instance, to discuss their codes and seek answers. As this student mentions:

> (. . .) what's surprised me the most is how . . . democratic is the wrong word but how many people can have an input in it and how many there are different languages or different . . . libraries there are and that kind of thing . . . So it feels more accessible that way, and I've learnt where to look for things and where to look for help (ID31, INT, female, 23 years old, English)

Learning with others was seen as more fun than taking a self-paced online course, and several students spoke of the rewards of interacting with a diverse body of students from different countries and disciplines: "it felt good that for once I didn't have to stay kind of within the confines of my subject and that I was encouraged to seek something out and do something new" (ID31, INT, female, 23 years old, English).

Although students were generally positive about working with others, it was highlighted, in the subcategory *group work management*, that to make collaborative work constructive, students needed a rationale for group work and explicit guidance for interaction. Overall, 19% of the students (N= 36) had the perception that the pairings had been random, and this felt less productive than working on their own. They commented that they would have preferred to choose who they worked with, and that having already built up a relationship made it easier to have learning conversations, as they were concerned about coming across "as like arrogant and that, I know what I'm doing" (ID27, INT, female, 19 years old, Economics/Business). Students' perceptions were that they had not really been advised about how to work together, and therefore the success of the group work seemed to depend on how "naturally" well they got on together. Groups that were not well equipped to divide tasks or to manage differences of opinion, when they didn't know each other, led to students feeling that the collaborative work had been wasted time. One student commented on how awkward it had been when no-one would speak and another mentioned that they learnt less because they "spent so much time balancing the various viewpoints of group members" (ID2, QUEST, female, 24 years old, Politics/International Relations). Indeed, negotiating viewpoints is an important skill in group work, but without it being highlighted as such students did not feel they had been learning. Nonetheless, overall students saw peer learning practices as important both for supporting their functional coding development and for raising awareness and understanding of the specifics of collaborating with code. The collaborative dimension of learning coding was therefore perceived as meaningful by students. As this student

puts it: "we can all code together and we come from different backgrounds and all work together. I thought that was super important" (ID17, INT, female, 22 years old, Politics/International Relations).

Contextualisation and critical & creative learning opportunities

Contextualisation practices were perceived as supporting wider awareness and understanding of the critical and creative aspects of working with code. The use of relevant *real-world resources*, for instance, allowed students to understand how learning could be applied in a range of different disciplines. Student responses to this aspect of the course were overall positive, with comments focusing on how the case studies made the learning tangible and engaging: "when we worked with the JSON files from the weather data, I could actually see the applications of persons that use sets of information provided by the Met Office or wherever and how it translates" (ID20, INT, male, 34 years old, Social Sciences). Students particularly appreciated when their tasks linked to a case study showing wider applications of a practice (*applied discussion*) as, for example, when the B2 workshop on ggplot was followed by a session on the use of graphs and visualisations in journalism. These sessions were designed to stimulate critical judgment based on data analysis. For example, highlighting questions around what makes a good sample, how to make good interpretations of regression analysis and how to draw well-founded conclusions. Students mentioned that they hadn't expected to learn "how important things like cleaning data are, that kind of thing. So just kind of valuing stuff that seems really abstract and you don't quite know what to do with when you just hear about it" (ID31, INT, female, 23 years old, English).

In general, this approach gave students a bigger picture and "more context of how you can apply it to your career and life" (ID10, INT, male, 21 years old, Politics/International Relations). The guest speakers sparked new ways of looking at potential career trajectories that they would not previously have been able to imagine or "didn't even know existed" (ID1, INT, female, 24 years old, History). The case studies also stimulated students to think more creatively about the possibilities afforded by coding: "he was doing amazing things. I would have not made the link between archaeology and coding" (ID17, INT, female, 22 years old, Politics/International Relations). In the group project one student commented that the lecturer had expressed how much creativity was needed, and this had resonated with her coming from a social science background:

> Just working on the group project and being able to think of a fun idea for us to do involved creative thinking and then it also involves a lot of problem-solving … if you have a set thing that you have to make, figuring out how to do it, so you have to be creative in how you go about undertaking a specific task (ID21, INT, female, 29 years old, Politics/International Relations)

In addition to contributing towards student understanding of how coding can be used in creative and interdisciplinary ways, the guest speakers' sessions, as well as the main lectures, were conceived as an opportunity to stimulate critical debate, which students in general appreciated. Comments in this regard expressed a better understanding of bias in data science and the societal impact of coding and AI, than they had prior to the course. As this student in B2 states:

> [For] arguments used in persuasive political debates, you always incorporate statistics and data and obviously it's refreshing to know that people who practice in statistics and work with data every day are aware of the biases around it and the dangers and how what's the difference between presenting things in a good way versus a bad way that (ID28, INT, female, 21 years old, Politics/International Relations)

Overall, the applied discussions helped students leave the course more critically informed about potential applications of coding, data science and AI: "I've got less of the sense that we're going to lose a lot of jobs to AI. I've got more of a sense that maybe it can create more jobs if we do it right in education" (ID37, INT, male, 34 years old, Education). However, in five of the 36 cases (14%), students found the guest lectures too intense without structured discussion. Finding ways of embedding critical and creative thinking through the workshop tasks is a challenge that still needs further exploration in studies of bootcamps like this. Nonetheless, the contextualisation practices mentioned were perceived of value for students in developing critical and creative thinking through and about coding.

Discussion and conclusion

The aim of this paper was firstly to characterise the pedagogic practices non-CS students identified as meaningful for learning with and about coding, and secondly to map these practices on to our existing framework to enhance its relevance and operational value

In terms of guided hands-on learning, our findings align with other computing pedagogy researchers who have shown that whilst students enjoy the opportunity to learn through trial and error, they like this to be supported by an appropriate framework and structured assignments (Luxton-Reilly et al., 2018). In this study, learners particularly referred to live coding as an effective and engaging instructional method. This has also been reported in the literature (Luxton-Reilly et al., 2018), along with other practices such as code tracing and sub-goal modelling, where labels are added to worked examples to visually group steps into sub-goals, thereby highlighting the structure of code (Waite, 2017). In the studied course, Jupyter notebooks were used and students identified their structure as being as a helpful way of decomposing learning into incremental steps. Building on this, it has been highlighted in another study that the cell-based structure of the notebooks can also be used as a way of developing students writing skills by improving their understanding of how to structure a report on a data investigation (Willis et al., 2020). Realising the potential of coding-specific resources to support multiple learning outcomes is one area for development.

In terms of peer learning, students in the course highly valued the way in which social bonding had been fostered. Other coding courses also report that building a sense of community and friendship is a key element, both so that students have peer support if they start to feel discouraged (Dawson et al., 2018) and to feel confident making and learning from mistakes (Chowdhury et al., 2018). Unlike in longer courses, the time to build trusting relationships was limited, but scheduling social activities proved to be rewarding. Some students also highlighted that they had gained an understanding of the teamwork involved in coding contexts. On the other hand, some students viewed the re-use of other people's code as less valuable than writing code themselves. This suggests that more could be done to raise explicit awareness of the value of working in collaborative, open-source environments

like GitHub, which allow coders to share code, comment on or remix other's work and contribute resources and new ideas (Burke et al., 2016; Jenkins et al., 2013). A key aspect for collaborative work to be effective is that students need training in order to successfully pair, for example, through introductory activities, explicit guidance around things like turn taking (Sanders et al., 2017), and through encouragement to reflect on the effectiveness of the pairing and report to the teacher when things are not working (Hanks et al., 2011). Our findings suggest there is a more central role for talk in supporting the learning of coding, acknowledging that a discursive approach develops not just communication skills but is fundamental to pushing conceptual, critical and creative thinking as well. In this, we agree with Khachatryan and Karst (2017) who argue that communication is a catalyst for analytic thinking, and ultimately a way of finding a solution to a problem (Khachatryan & Karst, 2017).

In terms of contextualisation, the use of real-world and discipline-specific examples and projects helped the students to engage, thus leading to an informed insight into potential career trajectories, and a more realistic understanding of useful skillsets. One student in our study realised coding was more creative than she had expected, while others realised there were intermediary or "broker" roles that would cater to their interests (Chilana et al., 2015). Several mentioned that the course had provided a roadmap and given them a clearer sense of purpose in terms of what they needed to learn. This addresses a gap identified in a study of bootcamps, which found that many potential students lack understanding of the broader skills that are actually useful in computing careers (Seibel & Veilleux, 2011).

Turning to the second contribution of this paper, Figure 1 shows how we have enhanced our framework using the findings from this study, by mapping the pedagogic practices and learning opportunities described against the dimensions previously established.

In attempting to map the practices identified with the learning opportunities described in the multidimensional framework, it should be noted that our findings do not suggest a simple match between pedagogic approach and dimension. Rather, they reveal those relationships as dynamic. For instance, although peer learning might be expected to support collaborative outcomes, we would argue that peer learning is also a significant factor in students' developing their functional understanding of code. Although

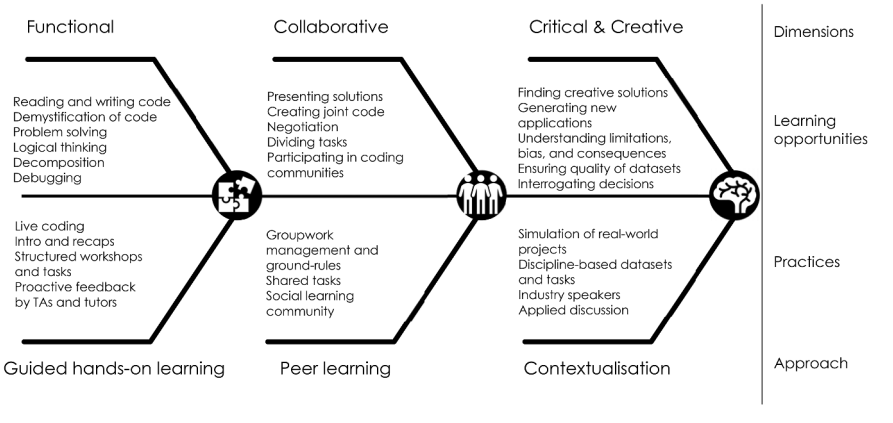

Figure 1. Identifying granular level practices to support learning opportunities across three dimensions

contextualisation, particularly through guest speakers' focus on social applications, might be expected to support critical & creative outcomes, it can also bring insight in terms of collaborative coding communities. Experimenting with aligning each strategy to different dimensions can therefore be a catalyst for creative thinking both about practices and learning opportunities.

By mapping the findings from this study onto a conceptual framework, we have shown that a range of learning outcomes in different dimensions can be supported through the practices highlighted. In particular, the findings suggest that in terms of supporting learners to understand how to use code in wider contexts, much can be gained by developing practices that guide interaction and thinking processes.

Limitations and future directions

Although this study builds on previous empirical work using a systematic review of the literature and uses mixed-method data, the fact that it adopts a case study methodology makes it less generalisable to other contexts. In addition, despite the value of emphasising student voice, triangulation with session observations and content analysis of outputs could provide more robust results in the future. Further work is therefore needed to consolidate the application of our framework to other contexts of teaching coding to non-CS learners.

As inclusive coding practices become increasingly important in workplaces and society, higher education institutions are facing an increasing demand for coding courses that appeal to a diverse range of learners. This research is therefore of interest to all higher education institutions, as they adapt to meet this need.

Notes

1. http://sites.exeter.ac.uk/instituteofcoding/
2. Since there were three questionnaires, one per block, the total number of the respondents regarding all blocks is $N = 56$.
3. The factor analysis followed the maximum likelihood method and rotation of Promax with Kaiser Normalization, having obtained a satisfactory sampling adequacy (KMO = 0.85, $p < .05$). Factor correlation = .5; Cronbach's alpha: F1 = .87; F2 = .83.
4. Approximately one TA to six or seven students, depending on the block.

Acknowledgments

This work was supported by the Institute of Coding (IoC) which received funding from the Office for Students (OfS) in the UK.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The author(s) reported that there is no funding associated with the work featured in this article.

Notes on contributors

Georgie Tarling is a Lecturer in the Graduate School of Education at the University of Exeter. Before joining the university, she was a television producer and then a teacher of English, Film and Media. Following her PhD, she worked as a postdoctoral research fellow for the Institute of Coding and the South West Institute of Technology. Her research interests lie in conceptualizing and developing pedagogies which support criticality and creativity in relation to digital practices. ORCID https://orcid.org/0000-0003-2699-005X

Ana Melro is a Lecturer in Communications at the University of Maia and a researcher in the Communication and Society Research Centre at the University of Minho (UMinho), Portugal. She holds a European PhD in Media and Education Studies from UMinho in partnership with the Autonomous University of Barcelona. She was a postdoctoral research fellow of the Institute of Coding, at the University of Exeter, where she studied the teaching and learning of coding. ORCID https://orcid.org/0000-0002-5649-4957

Judith Kleine Staarman is a Senior Lecturer in Education the Graduate School of Education at the University of Exeter, where she teaches and supervises on Masters (MA Technology, Creativity and Thinking) and doctoral courses. Her research interests include collaboration and classroom dialogue, both with and without technology, across curriculum strands and educational settings. She is also interested in new pedagogies for teaching, learning and thinking with technology. ORCID https://orcid.org/0000-0002-9504-1918

Taro Fujita is a Senior Lecturer in mathematics education in the Graduate School of Education at the University of Exeter. His current research interests include the history of mathematics education, the teaching and learning of geometry, mathematical thinking and reasoning, and the use of technology in mathematics education. ORCID https://orcid.org/0000-0002-3547-456X

ORCID

Taro Fujita http://orcid.org/0000-0002-3547-456X

References

Amaro, J. P., Barreiros, J., Coutinho, F., Durães, J., Santos, F., Alves, A., Silva, M., & Cunha, J. (2020). Challenges and solutions from an embedded programming bootcamp. *OpenAccess Series in Informatics*, *81*(2), 1–2. https://doi.org/10.4230/OASIcs.ICPEC.2020.2

Baker-doyle, K. J. (2018). I, Pseudocoder : Reflections of a literacy teacher-educator on teaching coding as critical literacy. *Contemporary Issues in Technology and Teacher Education*, *18*(2), 255–270 https://citejournal.org/volume-18/issue-2-18/english-language-arts/i-pseudocoder-reflections-of-a-literacy-teacher-educator-on-teaching-coding-as-critical-literacy/.

Barr, V. (2012). Create two, three, many courses: An experiment in contextualised introductory computer science. *Journal of Computing Science in Colleges*, *27*(6), 19–25 https://dl.acm.org/doi/abs/10.5555/2184451.2184458 .

Barr, V. (2016). Disciplinary thinking, computational doing: Promoting interdisciplinary computing while transforming computer science enrollments. *ACM Inroads*, *7*(2), 48–57. https://doi.org/10.1145/2891414

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, *87*(4), 834–860. https://doi.org/10.3102/0034654317710096

Burke, Q., Bailey, C., Lyon, L. A., & Green, E. (2018). Understanding the software development industry's perspective on coding boot camps versus traditional 4-year colleges. *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 2018*, January, 503–514. https://doi.org/10.1145/3159450.3159485

Burke, Q., & Bailey, C. S. (2020). Becoming an "adaptive" expert. *Communications of the ACM*, *63*(9), 56–64. https://doi.org/10.1891/9780826173157.0007

Burke, Q., O'Byrne, W. I., & Kafai, Y. B. (2016). Computational participation. *Journal of Adolescent & Adult Literacy*, *59*(4), 371–375. https://doi.org/10.1002/jaal.496

Camp, T., Adrion, W. R., Bizot, B., Davidson, S., Hall, M., Hambrusch, S., Walker, E., & Zweben, S. (2017). Generation CS: The challenges of and responses to the enrollment surge. *ACM Inroads*, *8*(4), 59–65. https://doi.org/10.1145/3141773

Carter, L. (2011). Ideas for adding soft skills education to service learning and capstone courses for computer science students. *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 517–521. https://doi.org/10.1145/1953163.1953312

Chilana, P. K., Alcock, C., Dembla, S., Ho, A., Hurst, A., Armstrong, B., & Guo, P. J. (2015). Perceptions of Non-CS Majors in Intro Programming: The Rise of the Conversational Programmer. *IEEE Computer Society* 251–259. https://ieeexplore.ieee.org/document/7357224

Chowdhury, B., Bart, A. C., & Kafura, D. (2018). Analysis of collaborative learning in a computational thinking class. *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education* 2018, January, 143–148. https://doi.org/10.1145/3159450.3159470

Cohen, J. D., Renken, M., & Calandra, B. (2017). Urban middle school students, twenty-first century skills, and STEM-ICT careers: Selected findings from a front-end analysis. *TechTrends*, *61*(4), 380–385. https://doi.org/10.1007/s11528-017-0170-8

Creswell, J. W. J., & Clark, V. L. P. (2006). *Designing and conducting mixed methods research*. Sage. https://doi.org/10.1016/j.aenj.2008.02.005

Czerkawski, B. C. (2015). Computational thinking in virtual learning environments E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education October 19, 2019 Kona, Hawaii. , (San Diego, CA: AACE), 1227–1231. https://doi.org/10.13140/RG.2.1.5173.8968

Davenport, J. H., Crick, T., Hayes, A., & Hourizi, R. (2019). the institute of coding: Addressing the UK digital skills crisis. *ACM International Conference Proceeding Series*, 0–3. https://doi.org/10.1145/3294016.3298736

Dawson, J. Q., Allen, M., & Campbell, A. (2018). Designing an introductory programming course to improve non-majors ' experiences. *SIGCSE'18: The 49th ACM Technical Symposium on Computer Science Education*, February 12–24 26–31.

Dempster, P., Onah, D., & Blair, L. (2020). Increasing academic diversity and inter-disciplinarity of computer science in higher education. CEP.

Falkner, K., & Sheard, J. (2019). Pedagogic approaches. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge handbook of computing education research*. (pp. 327). Cambridge University Press.

Grover, S., & Pea, R. (2017). Computational thinking: A competency whose time has come. *Computer Science Education: Perspectives on Teaching and Learning in School*, *19*(December), 19–39. http://hub.mspnet.org/index.cfm/33300

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: A literature review. *Computer Science Education*, *21*(2), 135–173. https://doi.org/10.1080/08993408.2011.579808

Jenkins, H., Clinton, K., Robison, A. J., Purushotma, R., & Weigel, M. (2013). *Confronting the challenges of participatory culture: Media education for the 21 century*. The MacArthur Foundation.

Kafai, Y. B., & Burke, Q. (2014). *Coding for all. In connected code: Why children need to learn programming*. MIT Press.

Kafura, D., Bart, A. C., & Chowdhury, B. (2015). Design and preliminary results from a computational thinking course. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE, 2015*, June, 63–68. https://doi.org/10.1145/2729094.2742593

Khachatryan, D., & Karst, N. (2017). V for voice: Strategies for bolstering communication skills in statistics. *Journal of Statistics Education*, *25*(2), 68–78. https://doi.org/10.1080/10691898.2017.1305261

Lehman, K. J., Doyle, M., Lyon, L. A., & Thayer, K. (2018). Alternative paths to computing careers and their role in broadening participation. *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 2018* January, 670–671. https://doi.org/10.1145/3159450.3159624

Lockwood, J., & Mooney, A. (2018). Computational thinking in secondary education: Where does it fit? A systematic literary review. *International Journal of Computer Science Education in Schools*, *2* (1), 41. https://doi.org/10.21585/ijcses.v2i1.26

Luxton-Reilly, A., Simon Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 55–106. https://doi.org/10.1145/3293881.3295779

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Lyon, L. A., & Green, E. (2020). Women in coding boot camps: An alternative pathway to computing jobs. *Computer Science Education*, *30*(1), 102–123. https://doi.org/10.1080/08993408.2019.1682379

Margolis, J., Fisher, A., & Miller, F. (1999). Caring about connections: Gender and computing. *IEEE Technology and Society Magazine*, *18*(4), 13–20. https://doi.org/10.1109/44.808844

McCord, R., & Jeldes, I. (2019). Engaging non-majors in MATLAB programming through a flipped classroom approach. *Computer Science Education*, *29*(4), 313–334. https://doi.org/10.1080/08993408.2019.1599645

Peteranetz, M. S., Soh, L. K., & Ingraham, E. (2019). Building computational creativity in an online course for non-majors. *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 442–448. https://doi.org/10.1145/3287324.3287346

Pollock, L., Mouza, C., Guidry, K. R., & Pusecker, K. (2019). Infusing Computational Thinking Across Disciplines. *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 435–441. https://doi.org/10.1145/3287324.3287469

Pon-Barry, H., Packard, B. W. L., & St John, A. (2017). Expanding capacity and promoting inclusion in introductory computer science: A focus on near-peer mentor preparation and code review. *Computer Science Education*, *27*(1), 54–77. https://doi.org/10.1080/08993408.2017.1333270

Sanders, K., Boustedt, J., Eckerdal, A., McCartney, R., & Zander, C. (2017). Folk pedagogy: Nobody doesn't like active learning. *ICER 2017 - Proceedings of the 2017 ACM Conference on International Computing Education Research*, October, 145–154. https://doi.org/10.1145/3105726.3106192

Santana, B. L., & Bittencourt, R. A. (2018). Increasing motivation of CS1 non-majors through an approach contextualized by games and media. *Frontiers in Education*.

Sax, L. J., Lehman, K. J., & Zavala, C. (2017). Examining the enrollment growth: Non-CS majors in CS1 courses SIGCSE '17 March, 2017 Seattle, WA. , *2017* (ACM), 513–518. https://doi.org/10.1145/3017680.3017781

Seibel, S., & Veilleux, N. (2011). Factors influencing women entering the software development field through coding bootcamps vs Computer Science Bachelor's Degrees. *The Journal of Computing Sciences in Colleges*, *26*(4). 84–96 https://dl.acm.org/10.5555/3344051.3344058

Soh, L. K., Samal, A., Scott, S., Ramsay, S., Moriyama, E., Meyer, G., Moore, B., Thomas, W. G., & Shell, D. F. (2009). Renaissance computing: An initiative for promoting student participation in computing. *ACM SIGCSE Bulletin*, *41*(1), 59–63. https://doi.org/10.1145/1539024.1508885

Stefan, M. I., Gutlerner, J. L., Born, R. T., Springer, M., & Fox, J. A. (2015). The quantitative methods boot camp: Teaching quantitative thinking and computing skills to graduate students in the life sciences. *PLoS Computational Biology*, *11*(4), 1–12. https://doi.org/10.1371/journal.pcbi.1004208

Tarling, G., Melro, A., Kleine Staarman, J., & Fujita, T. (2020). Coding for non-Computer scientists: Pedagogies for interdisciplinary participation. *Proceedings of EdMedia + Innovate Learning*, 581–586.

Vihavainen, A., Airaksinen, J., & Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success ICER '14, Proceedings of the 10th Annual Conference on International Computing Education Research July 2014 . , 19–26. https://doi.org/10.1145/2632320.2632349

Waguespack, L., Babb, J. S., & Yates, D. J. (2018). Triangulating coding bootcamps in is education: Bootleg education or disruptive innovation? *Information Systems Education Journal*, *16*(6). 48 http://isedj.org/2018-16/

Waite, J. (2017). Pedagogy in teaching computer science in schools: A literature review. *The Royal Society*, 1–90. https://royalsociety.org/~/media/policy/projects/computing-education/literature-review-pedagogy-in-teaching.pdf

Willis, A., Charlton, P., & Hirst, T. (2020). Developing students'written communication skills with Jupyter notebooks. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 1089–1095. https://doi.org/10.1145/3328778.3366927