Adv Mobile Learn Educ Res, 2022, 2(1): 187-200
DOI: 10.25082/AMLER.2022.01.003

RESEARCH ARTICLE

Strengthening the coding skills of teachers in a low dropout Python MOOC

Fotis Lazarinis[1*] **Anthi Karatrantou**[2] **Christos Panagiotakopoulos**[2] **Vassilis Daloukas**[2] **Theodor Panagiotakopoulos**[1]

[1] School of Technology and Science, Hellenic Open University, Patras, Greece
[2] Department of Education and Social Work, University of Patras, Patras, Greece

Correspondence to: Fotis Lazarinis, School of Technology and Science, Hellenic Open University, Patras, Greece; Email: fotis.lazarinis@ac.eap.gr

Received: December 21, 2021;
Accepted: January 2, 2022;
Published: January 6, 2022.

Citation: Lazarinis, F., Karatrantou, A., Panagiotakopoulos, C., Daloukas, V., & Panagiotakopoulos, T. (2022). Strengthening the coding skills of teachers in a low dropout Python MOOC. *Adv Mobile Learn Educ Res*, **2**(1), 187-200.
https://doi.org/10.25082/AMLER.2022.01.003

Abstract: In this paper, we present a structured approach to developing an outreach program aimed at improving the coding abilities of pre- and in-service teachers. The paper presents the design and development decisions made using the ADDIE model. External evaluators assessed the material's quality, confirmed the estimated workload, and examined the material's relevance to the educational goals. Learners' active participation was encouraged through multiple quizzes, and learners were assisted in their learning activities by means of practical examples. Based on the number of people who actually logged into the course, a completion rate of 70.84 percent is achieved. The paper presents and discusses the findings of an evaluation conducted with the assistance of experienced teachers and course participants.

Keywords: programming skills, coding, Python, teacher professional development, MOOC completion

1 Introduction

The core dimensions of Computational Thinking (CT) are analysis and design abilities, problem-solving skills, and logical thinking (ISTE, 2014). These skills are critical in today's fast-paced world, where the ability to adapt, deal with new challenges, and formulate structured solutions is a constant. Coding is a component of CT (Corradini et al., 2018), or a tool for promoting CT. It is the process that allows the solutions to be expressed in a computer encoded form, allowing the proposed logical steps to be tested. Coding is regarded as an important skill in today's society (Tuomi et al., 2018; Barr et al., 2011). Producing coding blocks will gradually assist individuals in understanding technical limitations, identifying potential flaws in their algorithmic thinking, and eventually allowing them to generalize their approaches. Coding also assists students in improving their cognitive abilities (Scherer et al., 2018).

At the same time, since digital technology is penetrating our daily lives, it is important to educate digitally literate citizens. Representing the disciplinary formalism of computer science, programming is considered a core practice, and this leads programming to the focus of computer science education (Krishnamurthi & Fisler, 2019). However, it is widely known that programming, by nature, is notoriously challenging for novices (Robins, 2019). Making an effective and engaging learning environment for novices has received a substantial interest (Lazarinis et al., 2019; Kim et al., 2018; Kwon, 2017). Therefore, it is critical to teach coding to pre-service and in-service teachers so that they can introduce coding and problem-solving skills to their students (Yadav et al., 2017). According to Rick et al. (2018), rather than hiring specialists to teach coding and computing, school principals primarily rely on already hired teachers to teach computer science topics. Some projects use primarily face-to-face sessions to accomplish this (Lamprou et al., 2017); others use block-based languages in e-learning settings; and still others use text-based languages (Klimeková & Tomcsányiová, 2018; An & Lee, 2014).

The terms CS, CT, programming, and coding are frequently used in CS research. The last three terms are sometimes used interchangeably which is incorrect. CS encompasses the entire scientific field, including all subfields such as hardware, software, networks, databases, etc. CT is primarily concerned with algorithmic problem-solving skills. Coding is the process of converting natural language into machine commands via an intermediary coding language. Coding is a subset of programming, and both are needed to develop CT. Coding is the preferred method in school education, where the goal is to help students think more algorithmically. Students and teachers code in block and text-based languages to solve short problems.

In this paper we present the motivation, design decisions, implementation details, and evaluation of a University outreach program. The remainder of the paper is organized as follows: in section 2, we review papers on computational thinking in order to establish its importance

for school students and teachers. We also look at papers on which programming language to teach, what approach to take, and which medium to use to reach out to potential students more effectively. Section 3 presents the research objectives, design decisions, and implementation details for the Python MOOC (Massively Open Online Course). Section 4 is devoted to course evaluation, and the final section concludes the paper.

2 Literature review

Facilitating the incorporation of CT into school education is a common goal shared by both researchers and practitioners. Some of the initiatives promoting CT in schools include the Hour of Code (https://hourofcode.com), Google's programs (https://edu.google.com/code-with-google), and the Wolfram Foundation's Computational Thinking Initiatives (https://www.computationinitiative.org).

Developing CT and learning to code have specific benefits in terms of student engagement, motivation, confidence, problem-solving, and communication (Scherer et al., 2018). Appropriate interventions are required to improve teachers' problem-solving abilities, self-efficacy, attitudes, and knowledge (Poultsakis, 2021; Mason & Rich, 2019). According to a Google and Gallup report, there is a positive growth in computer science (CS) classes in the United States, with more principals reporting that their school offers a CS class with programming or coding. Furthermore, the study demonstrates that key concepts, such as CT, are being integrated into classrooms. A large percentage of parents (84%) and principals (66%) believe that offering CS is more important than or equal to required courses such as math, science, history, and English.

CT in schools research focuses on novel approaches to preparing students and teachers to be a part of this new reality. CT development approaches include data collection, analysis, and representation, problem decomposition, the use of algorithms and procedures, and simulation (Gretter & Yadav, 2016). A review of the literature was conducted to synthesize research on pre-service and in-service programs that improve K–6 teachers' attitudes, self-efficacy, or knowledge of coding and CT (Mason & Rich, 2019). The study's findings suggest that training that encourages active participation can boost teachers' computing self-efficacy, attitudes, and knowledge. Semi-structured interviews with Australian teachers revealed that participants had varying levels of CT competence. According to the study, coding and CT are still relatively new topics that should be integrated into the studies of new teachers in order to strengthen their skills (Lloyd & Chandra, 2020). Several obstacles may stand in the way of educators successfully teaching elementary computing. Low self-efficacy or a lack of technological, pedagogical, or content knowledge may prevent them from teaching computing successfully (Ertmer et al., 2012). Teachers believe that CT entails logical thinking, calculation, and problem solving (Sands et al., 2018). Educational robotics e-courses have been implemented for teachers (e.g. Tzimopoulos et al., 2021).

Coding is the primary method for involving individuals in CT. In contrast to CS unplugged (Bell & Vahrenhold, 2018) activities that use games and puzzles to help students and teachers develop CT skills, coding involves analysis, data organization, writing in some type of machine language, and assisting students in testing their solutions. It promotes learning through active participation and motivating students who can see the outcomes of their actions. Coding activities are frequently manifested through the use of block-based languages. Scratch and Alice, two visual block programming tools, have been used to introduce coding to teachers (Vaca-Cárdenas et al., 2020). Although these approaches are beneficial and more accessible to newcomers to coding, they may be less effective when the goal is to make the skills transferrable to real-world problems. Those who learn to write code in a more common programming language, both graphically and on the console, will find it easier to transition to a more advanced computer programming language with even more complicated syntax (Weintrop & Wilensky, 2017). According to this study, both block-based and text-based languages can introduce CT to students with text-based languages in order to better support students' progression to more complicated languages.

Python is a programming language with a short and clean syntax, enforced structural design, and dynamic typing. Python has already been used in schools to teach coding, and its benefits have been identified (Grandell et al., 2006). Python is considered simpler than more complicated languages such as C++ due to its pseudocodish syntax and higher abstraction (Kunkle & Allen, 2016; Ateeq et al., 2014), making it more suitable for school education. A visual extension of Python was successfully used in a workshop for high school science teachers (Ahamed et al., 2010). When Java and Python were compared for teaching programming in (Mannila et al. 2006), it was discovered that students produced fewer errors in Python due to its lower syntactic complexity. Python was discovered to be one of the easiest and most popular programming

languages for students and teachers to learn (Noone & Mooney, 2018). Python, in the opinion of experienced educators, meets the majority of the challenges encountered when programming is introduced in secondary education (Mészárosová, 2015). Secondary education teachers' experiences with Python in formulating solutions in inquiry-based teaching approaches are similar (Guni et al., 2020).

Professional development programs for teachers are delivered in the form of face-to-face workshops or via distance learning. CT topics, for example, have been taught to students and teachers in Switzerland through intensive face-to-face sessions and complimentary online materials (Lamprou et al., 2017). A coding and CT MOOC for Finnish primary school teachers has been implemented, with the primary goal of teaching teachers computational thinking and basic programming concepts (such as commands, loops, and conditional statements) (Toikkanen & Leinonen, 2017). The e-course was built on a Moodle platform, with embedded Google forms, YouTube videos, and Padlets for collaboration. The course covered Scratch and ScratchJr, and the initiative received positive feedback. In this article, we will look at a Scratch blended learning course for teachers. The study describes the design decisions, topics, and organizational structure of the blended approach that resulted in lower dropout. Some of the techniques proposed in this work are also applicable in the current study, though the more demanding teaching materials and the longer duration necessitate the use of alternative techniques to assist teachers. MOOCs have been incorporated into teacher professional development programs (Spradling et al., 2015; Kellogg & Edelmann 2015). MOOCs should be long-term in order to provide teachers with a long-term learning opportunity (Hodges et al., 2016). Dropout prevention and prediction methods are described in two recent studies (Panagiotakopoulos et al., 2021; Kostopoulos et al., 2021) but designing effective courses with alternative options are required to support higher completion rates.

According to the review of studies, coding is an important aspect of CT. There are some advantages to using Python to implement coding activities. The added value is more obvious because it is a popular and professional programming language with a clean and simple syntax that reduces the learning curve. Progressing gradually from simple examples that help the learner understand the syntax of the language to more complex problems improve their analysis and problem-solving skills.

3 The GPython MOOC

Using an xMOOC (Kesim & Altinpulluk, 2015) as the realization path is deemed necessary to impact as many learners as possible, especially during these challenging times. For effectively advancement of the professional development (PD) of teachers, the key features are content focus, active learning, coherence, duration, and collective participation (Desimone, 2009). The main research question is "*does the participation in GPython MOOC improves the coding abilities of teachers?*".

Based on the literature review, there is a need to educate teachers in coding who in turn will help their students. In this work we follow a structured approach to develop an e-course. Formulating a robust design and development team helps in improving the design and deployment of the course and eventually to reduce the drop our rates and improve the actual skills of the participants. These are some of the novelties of our work. Below we analytically present all the design decisions as a guide for MOOC designs.

3.1 Applying the ADDIE MODEL to design and implement the e-course

ADDIE (Analysis, Design, Development, Implementation, Evaluation) is an instructional design method used as a framework in designing and developing educational and training programs (Branch, 2009). The last three phases, are run on short iterative steps to speed up the time between the design of the system and the delivery of parts of the course. (see Figure 1)

3.1.1 Analysis

In this first phase of the project we focused on the target audience, their characteristics, the possible constraints, the timeline of the project, the delivery options, and the main objectives of the project. We also dealt with the required roles of the development team, so as to ensure that the project will be deployed as planned.

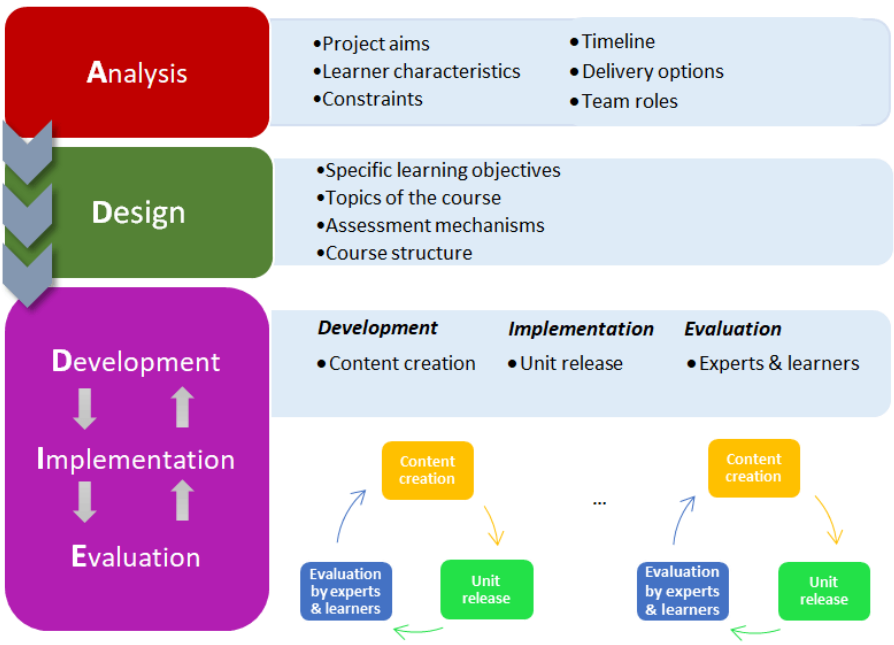

Figure 1 Application of the ADDIE model in a semi-linear and iterative approach

The main target of our project was to advance the coding abilities of school teachers by providing certified training to at least 300 teachers. The high-level project aims are summarized in the following list:

(1) To enhance the problem-solving skills of school teachers;

(2) To strengthen their algorithmic thinking;

(3) To support the participants to learn Python by using a practical approach;

(4) To produce online teaching materials matching the Greek high schools' curriculum.

The target learning group of our University department and research cluster is pre-service and in-service teachers. To participate in the course, teachers did not need prior coding experience. In Greece, programming languages are only taught in the last two classes of vocational high schools. Hence, we needed to teach coding to teachers who will then teach their students through formal or informal teaching activities.

Additional key decisions of the project were to deliver the course via distance learning for 60 hours over 12 weeks. The literature shows that a teacher PD program should be moderately long. Deploying the project over three months will also provide a balanced workload of approximately 5 hours per week, reducing teacher burnout.

An important decision concerned the formation of the development team, i.e. the required key roles. Based on our experiences with distance learning education and by studying practical guides for developing e-courses, we identified the following roles, in addition to the *Project Manager*:

(1) Instructional designer: analyses the needs and provides consultation on instructional strategies;

(2) Subject matter expert: provides content, resources, expert advice and checks the accuracy of the materials;

(3) Editor: reviews the materials for clarity, consistency, syntax and spelling errors;

(4) Content/multimedia developer: edits the multimedia content, e.g., images, videos, podcasts, et al.;

(5) Information technology expert: deals with IT topics, such as account set up, installment of development applications, and of other communication and information sharing channels, et al.;

(6) e-Learning developer: implements the e-course realizing the instructional designer decisions and the advices of the subject experts;

(7) Testers (quality assurance): they follow the e-course, assuring that it works correctly and the it meets the objectives;

(8) Educators/Facilitators: they support the learners during their interaction with the e-course.

All of the above categories are required for the project's success. Low motivation, lack of time, and lack of support are all factors that contribute to high dropout rates in MOOCs

(Eriksson et al., 2017). Insufficient learner support reduces student completion rates. Interaction between peers and students and educators can improve student engagement (Gregori et al., 2018). We identified the importance of educators/facilitators who will support the learners via email and forums. These retention strategies will improve peer and learner communication.

3.1.2 Design

The second stage of the model identifies the specific learning objectives, topics, assessment mechanisms, and overall course structure. We first decided on the registration and teacher recruitment procedures. This would be achieved by posting on educational websites and networking with teacher associations.

The high level aims of the previous phases have been manifested to specific objectives:

(1) to realize what an algorithm is;
(2) to be familiar with the basic commands/structure of an algorithm;
(3) to install and utilize the Python development environment;
(4) to comprehend the role of variables, input and output commands in Python;
(5) to develop short sequential programs in Python;
(6) to develop programs using selection commands in Python;
(7) to realize the importance of iteration and to form solutions to problems using selection and iteration commands in Python;
(8) to use strings in Python;
(9) to utilize lists, tuples, dictionaries, sets data structures in Python;
(10) to develop programs with built-in and user created functions in Python;
(11) to import basic Python modules in a program;
(12) to be able to download and use new Python modules;
(13) to read and write data in text files in Python;
(14) to employ the Python format function to format output;
(15) to utilize lists as stacks and queues to solve conventional problems;
(16) to code common sorting and searching algorithms in Python;
(17) to utilize the sorting and searching functions of Python;
(18) to introduce advanced Python capabilities.

The objectives were quantified based on the project's high-level goals. The first three high level goals aided in defining the learners' desired skills and the last in forming the curriculum. Based on the demand for a 12-week course, the course units are:

(1) Solving problems with computers. Introduction to algorithms;
(2) Introduction to Python;
(3) Solving problems with selection;
(4) Solving problems with iteration (1);
(5) Solving problems with iteration (2);
(6) Processing strings;
(7) Introduction to data structures;
(8) Solving problems with data structures;
(9) Advanced data structure topics;
(10) Functions and modules;
(11) Text files and output format;
(12) Sorting, searching and advanced topics.

Each unit is divided into two or three subunits. The material is presented in increasingly difficult problems. The decision to create units with comparable workloads was critical so that learners could immediately see the weekly workload. Topics in each unit are taught by using lots of practical examples. The rich variety of examples aims for the course to be useful for a novice in programming to get comfortable with coding, and not boring for an experienced programmer.

Finally, the evaluation mechanisms were determined during the design phase. Users were required to pass 80 percent of the tests without making any errors. They could repeat the test twice. Closed-ended items, such as multiple choice and fill-in-the-gap questions, were used in testing. Learners were required to type and run programs so as to select the correct answers since the copy function was inactive on the browser.

3.1.3 Development

For each unit and subunit, we created the Moodle course structure and the materials (aims, prerequisites, expected results, learning material and quizzes) (e.g. Figure 2 and 3). The materials were then uploaded to the course and evaluated by external testers. The team could

then move on to the next unit and the ready material was then released to the learners and educators/facilitators.

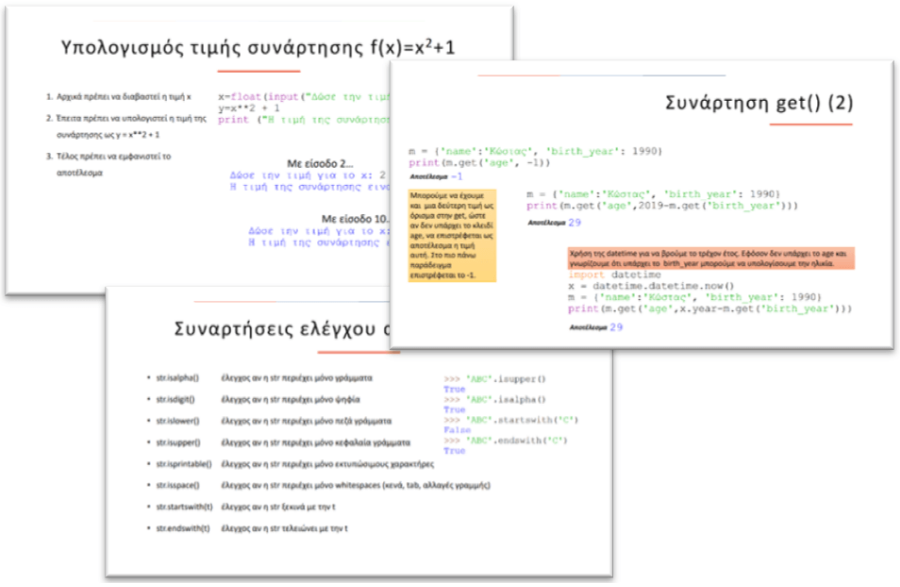

Figure 2 Examples of learning material (in Greek)

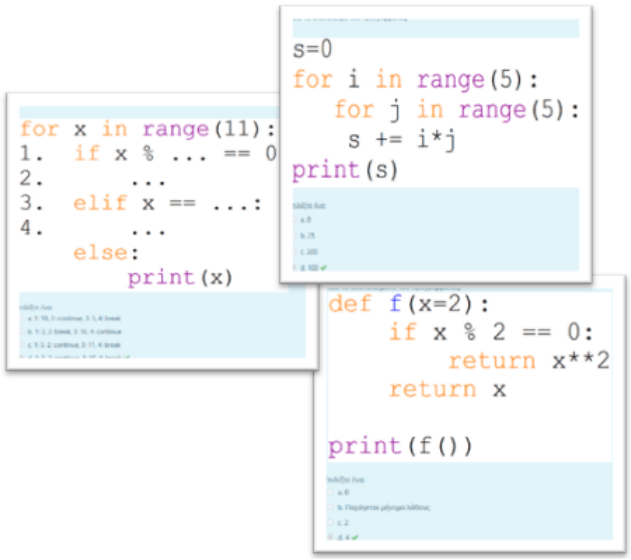

Figure 3 Examples of assessment items

3.1.4 Implementation

During this stage, the course units are introduced to the target audience and the learning process starts. The instructions and the rules to the participants are posted and the registration forms are made available to participants (see Figure 4 and 5). Training facilitators support the learners through the forum and by email if needed. They also check the reports of the Moodle to evaluate the progress of learners. Specific reminders are sent to the learners.

3.1.5 Evaluation

The first screening process of each unit is performed by the designated testers who are experienced Python educators and with solid backgrounds in distance learning. They test the material for errors and inconsistencies, and they follow the online course as regular learners. Their purpose was to review the process, to estimate the needed time, to attest that the objectives

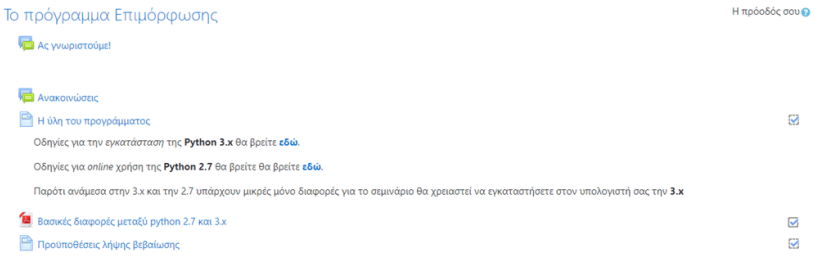

Figure 4 Requirements and introductory notes of the course (in Greek)

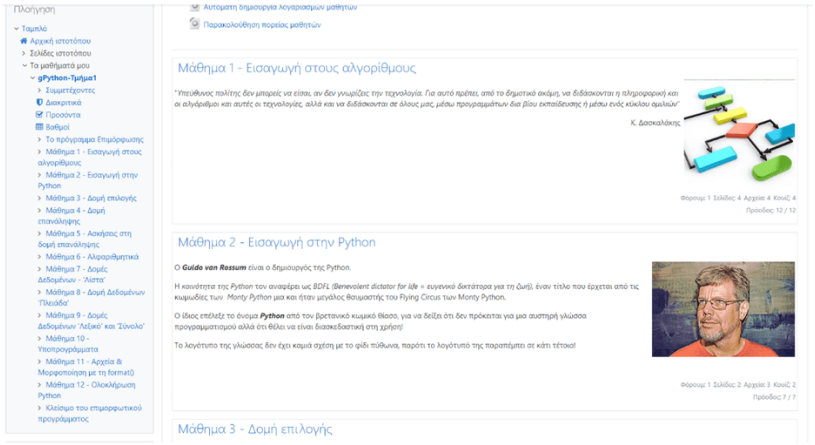

Figure 5 Implementation of the course in Moodle (in Greek)

are met, and to examine the accuracy of the assessment items as entered in the Moodle. They thus facilitate the spinning of the "development-implementation-evaluation" circle.

The second source of feedback comes from the trainers/facilitators. They examine the reports of the LMS, they regularly examine the forum and the interaction of the learners and communicate potential problems to the development team.

The learners themselves are the last, and clearly the most important, source of evaluation data. Indirectly, with their questions in the forum, their engagement with the learning material, and their performance to the tests, they provide useful data. These data can be exploited through various analytics plugins available for Moodle. Directly, through end-of-course summative questionnaires learners can provide useful reflections.

3.2 Course deployment

The course was implemented in two rounds, with the goal of having 300-400 registrations per sequence, resulting in a maximum of 200-300 actual participants per round. The actual number of participants had to be within the above limits in order for us to effectively manage the process and support the participants. The first cycle of the course runs concurrently with the course's development.

There was an open call for pre-service and in-service teachers, with the requirement to register by filling out a registration form. The call was for both rounds, and teachers had to fill in their current Python knowledge, teaching specialty, gender, and preferred participation round. These details were critical in the event that we received more registrations than we needed and had to choose teachers based on their specialty, Python skills, and gender in order to maintain a balanced group of participants.

The project was launched in September 2019, but due to bureaucratic constraints, the team formation and actual work on the project could not begin until the beginning of October 2019. The first round of the course began in January 2020, and the second round began at the end of February 2020. We consider this a quick process because we had to prepare learning and assessment materials for 12 Python units, implement the MOOC, and manage the entire process with a small team of seven people and two external testers. The majority of team members took on the responsibilities of two or three team roles. For example, subject matter experts also served as educators, and multimedia developers aided in the development of MOOCs, thereby supporting all of these subtasks. The external testers were only in charge of the critical work of

testing the content and implementing the course in Moodle.

4 Evaluation of the course

4.1 Methods

To address the main research question, the main dimensions of the evaluation focused on:
(1) the perceived relevance and quality of the training activity;
(2) the participants' learning acquisition.

These variables were adapted from the Online Student Engagement Scale (Dixson, 2015), as in (Pérez-Foguet et al., 2018). The first dimension is evaluated based on expert opinions and the completion rate, which is a clear indicator of the course's relevance to the needs of the participants. In addition, the actual participants' responses to the following eight questions have been recorded:

Q1. The objectives of each module were supported by the respective educational material
Q2. The content was well organized into sections
Q3. The educational materials were presented in a clear and comprehensive way
Q4. There were some prerequisites that made it difficult to attend the program
Q5. The coding examples were clear, detailed and could they be easily executed
Q6. The quizzes were clear and understandable
Q7. The quizzes corresponded to the content of each section
Q8. Overall, how do you evaluate the educational material of the program, in terms of its quality?

The second dimension is assessed based on the responses of the participants to the questions:

Q9. I knew how to program in a programming language prior to the PD program
Q10. I knew how to program in Python prior to the PD program
Q11. My expectations from participating in the program have been met
Q12. Are you feeling comfortable in teaching Python in your classes?

4.2 Evaluation by experienced teachers

The external testers were experienced computer science educators. Both had over 15 years of teaching experience, including time spent teaching Python in vocational secondary schools. Prior to working together, they had both taken some distance learning courses, either self-paced in Moodle or in synchronous mode. As a result, they had the necessary experience to evaluate our efforts. The procedure is divided into the following steps:

(1) They examined the individual materials of each unit for errors, ambiguities, and inconsistencies between the learning materials and the assessment items;

(2) They followed the individual units in Moodle as learners so as to have a genuine learning experience and to use the support services (forum, email, announcements). That way they could appreciate the true requirements and the workload;

(3) Last, they had to produce a report for every unit citing potential errors and ambiguities, their workload estimation, the difficulty level and the suitability of the material for novices in programming.

Both testers discovered simple errors (typos, syntax errors, some wrong numbers, ambiguities in textual explanations). Only one to two coding errors were reported per unit. They both agreed that the course is appropriate for beginners in coding and that by following the course, they will improve their abilities and gain a solid understanding of Python programming. They stated that the materials are of increasing difficulty and that they are indeed in accordance with the objectives of each unit. They suggested that for only one unit, one or two testing items to be added to cover a portion of the learning material that was not covered in the quizzes.

4.3 Registrations and completions

Table 1 shows the demographics of the pre-service and in-service teachers who expressed an interest in our course. The demographics show a distribution among specialties and sexes, with computer science teachers having a higher registration rate. This is considered natural because the program is primarily concerned with their specialty and teaching activities. It could be argued that the number of registrations from CS teachers demonstrates the need for focused programming PD programs.

Table 2 shows the number of participants who logged in and completed at least the first quiz of the course. These people are considered to be the actual participants in our course.

Table 1 Registrations of teachers for both rounds

	Pre-service Teachers		In-service Teachers		Total (n, %)
	Female	Male	Female	Male	
CS	20	29	149	148	346 (55.27%)
NS	48	29	25	34	136 (21.73%)
SH	15	2	81	46	144 (23.00%)
Total (n, %)	83 (58.04%)	60 (41.96%)	255 (52.7%)	228 (47.3%)	626(100.00%)

Note: CS: Computer science; NS: Natural science; SH: Social sciences/Humanities

In total, 559 teachers accessed the MOOC and took at least one quiz. These teachers will be used to calculate the completion rate, which is a strategy that has already been proposed in the literature (Ho et al., 2015) and is known as the "effective dropout rate" (Huin et al., 2016). Nearly 55% of in-service teachers work in secondary education, while the remainder work in primary, post-secondary, or provide administrative or counseling services during the current school year.

Table 2 Participants who completed at least the first quiz of the course

	Pre-service Teachers		In-service Teachers		Total (n, %)
	Female	Male	Female	Male	
CS	17	25	129	140	311 (55.64%)
NS	40	27	23	32	122 (21.82%)
SH	10	1	74	41	126 (22.54%)
Total (n, %)	67 (55.83%)	53 (44.17%)	226 (51.48%)	213 (48.52%)	559(100.00%)

Note: CS: Computer science; NS: Natural science; SH: Social sciences/Humanities

The number of teachers who completed the course and received a certificate of achievement is shown in Table 3. Overall, 396/559 teachers (70.84%) completed the course. Even if success is defined in terms of Table 1 data, the completion rate exceeds 63%, which is quite high for MOOCs.

Table 3 Participants who received the certificate of achievement

	Pre-service Teachers		In-service Teachers		Total (n, %)
	Female	Male	Female	Male	
CS	9	21	100	115	245 (61,87%)
NS	28	12	24	20	84 (21,21%)
SH	10	1	33	23	67 (16,92%)
Total (n, %)	47 (58.02%)	34 (41.98%)	157 (49.84%)	158 (50.16%)	369(100.00%)

Note: CS: Computer science; NS: Natural science; SH: Social sciences/Humanities

4.4 Evaluation by learners

The survey of user opinions focused on the course's organization, the material's quality, and the learning gain. The survey was conducted by the 396 students who completed the course at the conclusion. At the conclusion of the course, students were required to complete a mandatory questionnaire consisting of questions with responses on a 5-grade Likert scale. A team of three experts reviewed the questionnaire's content validity (in ICT in education, in programming and in distance learning). A pilot study was conducted with four (4) teachers to determine the questionnaire's face validity.

Due to the absence of statistically significant differences in the responses of participants in the two rounds, we report the results of the two rounds cumulatively. The questionnaire was completed by the 81 pre-service and 315 in-service teachers who took the test (51.52 percent female, 48.48 percent male). Nearly 93 percent of respondents held a university degree, and a sizable proportion held a postgraduate degree or held a PhD. Following the demographic questions, participants were required to respond to a series of questions that were logically classified into four distinct categories.

The data were analyzed descriptively and explanatorily using the SPSS statistical package. The χ^2-test of independence was used to detect statistically significant differences and correlations between the variables in the study based on the type of data for each variable. Cronbach's

internal consistency coefficient was used to determine the reliability of the questionnaire responses, and it was found to be acceptable ($\alpha = 0.79$). The findings are presented for all teachers because no statistically significant differences between pre-service and in-service teachers were observed ($p > 0.05$ for all questions).

The following section of the questionnaire inquired about the course's content and objectives, which are associated to the relevance and quality of the course. As shown in Table 4, the majority of participants expressed favorable attitudes toward the course. The vast majority believed that the content and quizzes supported the objectives. The instructional materials were presented in a logical and structured manner. The issue of prerequisites is the only one that requires additional investigation in order to improve the course.

Table 4 Questions about the content

	Disagree or Strongly disagree			Neutral			Agree or Strongly agree		
	CS	NS	SH	CS	NS	SH	CS	NS	SH
Q1	0.41%	0.00%	1.49%	6.94%	9.52%	10.45%	92.65%	90.48%	88.06%
Q2	0.00%	0.00%	2.99%	4.08%	8.33%	7.46%	95.92%	91.67%	89.55%
Q3	0.00%	0.00%	0.00%	4.08%	8.33%	19.40%	95.92%	91.67%	80.60%
Q4	63.27%	60.71%	46.27%	29.80%	29.76%	38.81%	6.94%	9.52%	14.93%
Q5	0.00%	0.00%	0.00%	3.27%	10.71%	29.85%	93.47%	89.29%	82.09%
Q6	0.00%	0.00%	2.99%	7.35%	10.71%	22.39%	92.65%	89.29%	74.63%
Q7	0.00%	0.00%	1.49%	6.94%	9.52%	7.46%	93.06%	90.48%	91.04%

Note: CS: Computer science; NS: Natural science; SH: Social sciences/Humanities

This part of the questionnaire contained the concluding question "*Q8. Overall, how do you evaluate the educational material of the program, in terms of its quality*?" with 1-10 numerical rating (1: very unsatisfying; 10: excellent). More than 85% from each group of teachers ranked the quality of the material from 8 to 10. Within the CS teachers, this rating was favored by more than 90%. The rest 10-15% of the participants assessed the material with a mark 6 or 7.

Higher percentages for CS teachers and NS teachers (95.92% for the CS teachers and 91.67% for the NS teachers) than for SH teachers (80.60%) agree or strongly agree that the educational materials were presented in a clear and comprehensive way while a percentage of 19.40% for the SH teachers is neutral. The differences are statistically significant ($\chi^2(8) = 24.62$; $n = 396$; $p < .01$). Most teachers (CS, NS and SH teachers) disagree/strongly disagree that there were some prerequisites that made it difficult to attend the program. A percentage of 38.81% for the SH teachers expresses a neutral point of view and a percentage of 14.93% agree or strongly agree that there were some prerequisites that made it difficult to attend the program (statistically significant differences - $\chi^2(8) = 17.51$; $n = 396$; $p < .05$). Although most teachers (CS, NS and SH teachers) found that the coding examples were clear, detailed and could they be easily executed, a percentage of 29.85% for the SH teachers expresses a neutral point of view (statistically significant differences - $\chi^2(8) = 45.49$; $n = 396$; $p < .001$). A higher percentage of CS teachers and NS teachers (92.65% for the CS teachers and 89.29% for the NS teachers) than for SH teachers (74.63%), agree or strongly agree that the quizzes were clear and understandable while a percentage of 22.39% for the SH teachers is neutral (statistically significant differences - $\chi^2(8) = 21.81$; $n = 396$; $p < .05$). There were no statistically significant differences among CS, NS and SH teachers as the majority of them agree or agree strongly that the objectives of each module were supported by the respective educational material, the content was well organized into sections and the quizzes corresponded to the content of each section ($p > 0.05$).

The next part related to the learning gain and the satisfaction of the participants. From Table 5 we can conclude that most CS teachers knew how to program at least in one programming language. However, the rates are reverse in the case of Python. The other categories of teachers had significantly lower coding skills in general and in Python more specifically. However, most of the teachers agreed/strongly agreed that their expectations have been met.

Table 5 Questions about the prior knowledge and the expectations

	Disagree or Strongly disagree			Neutral			Agree or Strongly agree		
	CS	NS	SH	CS	NS	SH	CS	NS	SH
Q9	4.90%	34.52%	82.09%	34.29%	44.05%	10.45%	60.82%	21.43%	7.46%
Q10	48.16%	58.33%	88.06%	33.88%	32.14%	8.96%	17.96%	9.52%	2.99%
Q11	0.00%	2.38%	2.99%	14.69%	13.10%	26.87%	85.31%	84.52%	70.15%

Note: CS: Computer science; NS: Natural science; SH: Social sciences/Humanities

Most of the SH teachers (82.09%) did not know how to program in a programming language

prior to the PD program, contrary to the 60.82% for the CS teachers who knew how to program and the 65.48% of the NS teachers who were neutral or knew how to program (statistically significant differences ($\chi^2(8) = 215.35$; $n = 396$; $p < .001$). Most of the SH teachers (88.06%) did not know how to program in Python prior to the PD program contrary to the 48.16% for the CS teachers who had no programming experience with Python and the 58.33% of the NS teachers (statistically significant differences - $\chi^2(8) = 53.88$; $n = 396$; $p < .001$). There were no statistically significant differences among CS, NS and SH teachers as most of them agree or agree strongly that their expectations from participating in the program have been met. The percentage of the SH teachers who agree or agree strongly that their expectations from participating in the program have been met is lower than the respective percentages for the CS and NS teachers as the SH seem to face some difficulties during the program but the differences were no statistically significant ($\chi^2(8) = 12.38$; $n = 396$; $p > 0.05$).

To validate that the learners could indeed use Python, the section contained the question "*Q11. Are you feeling comfortable in teaching Python in your classes*?". More than 84% of the participants replied "*yes*" and the rest was divided between "*maybe*" and "no". This is a convincing indication that learners developed coding skills in Python as they felt confident to teach their students. The teachers could utilize the presentations and all the material of the course in their classes. The participants of our PD program had to complete approximately 80% of the quizzes with no error and a final assessment. Thus, they had to complete at least 23 of the total quizzes and to execute Python codes, which reinforced their real skills.

5 Conclusions

The purpose of this study is to describe the design decisions and implementation details for an e-course aimed at improving pre-service and in-service teachers' coding abilities. The literature review established the importance of teachers to develop their coding skills and, over time, their computational thinking. Python was chosen for its simplicity and steeper learning curve, and a MOOC was chosen for delivery of the course. The course promoted active learning through practical examples, assignments and tests, while maintaining a manageable workload and duration. The course was designed and developed in short iterative cycles to minimize the time required to deploy the course. We discuss all design decisions in detail in the paper and provide specific implementation details so that the work can serve as a model for similar courses.

Our primary objective was to assist teachers in enhancing their coding skills. According to the experts' and participants' responses, the specific MOOC assisted teachers in improving their coding abilities. Teachers learned to code and were required to confront problems in a series of ordered steps that required them to analyze and organize data in structures and to implement solutions in Python. The trainees unanimously agreed ($> 84\%$) that they are confident in their ability to use Python in their classes. The relevance and quality of the training activity, as well as the participants' acquisition of knowledge, have been validated through participant responses and increased completion rates.

From a total of 626 registrations, 559 teachers worked on at least one course's test and 396 completed the entire course. The completion rate is 70.84 percent for active participants and 63% based on the total registrations. In either case, the completion rate is significantly higher than the average MOOC completion rate reported. In our work, we attempted to address the factors that contribute to increased dropout rates more effectively. The course's requirements were made explicit from the start; students received continuous support throughout their training via forum and email communications; the student-to-teacher ratio was increased; and the course was directly connected to the high school curriculum, which increased participant motivation. Additionally, participants received a certificate of attendance, which serves as an additional motivator.

The course contains valuable information for learners new to computing, but at the same time it could be an interested course for the experienced programmer who wants to learn a new programming language and become a more skillful programmer. Thus, it could be an effective bridge between an introduction to programming and an advanced course on computational science.

In comparison to other approaches, coding appears to better meet this need while also assisting individuals in acquiring quantifiable skills. Practical training activities that are directly related to their job responsibilities appear to be more motivating for adult learners. Our work could be expanded by developing additional non-mandatory learning materials to better meet the needs of users with advanced programming backgrounds. Peer communication and support could be facilitated even more in order to boost collaboration. Focused interviews with various

user categories, including those who did not complete the course, are necessary to gain a better understanding of their needs for the subject at hand and for e-learning in general, as well as to improve the effectiveness of future professional development programs.

References

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Early, J. P. (2010). Computational thinking for the sciences: A three-day workshop for high school science teachers. Proceedings of the 41st ACM technical symposium on Computer science education (pp. 42-46). Milwaukee, Wisconsin, USA: ACM.

An, S., & Lee, Y. (2014). Development of Pre-service Teacher Education Program for Computational Thinking. In M. Searson & M. Ochoa (Eds.), Proceedings of SITE 2014–Society for Information Technology & Teacher Education International Conference (pp. 2055-2059). Jacksonville, Florida, United States: Association for the Advancement of Computing in Education (AACE). Retrieved December 7, 2020.
https://www.learntechlib.org/p/131092

Ateeq, M., Habib, H., Umer, A., & Ul Rehman, M. (2014). C++ or Python? Which one to begin with: a learner's perspective. In International Conference on Teaching and Learning in Computing and Engineering (LaTiCE 14). IEEE, 64-69.
https://doi.org/10.1109/LaTiCE.2014.20

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. Learning & Leading with Technology, 38(6), 20-23.

Bell, T., Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work?. In: Böckenhauer HJ., Komm D., Unger W. (eds) Adventures Between Lower Bounds and Higher Altitudes. Lecture Notes in Computer Science, vol 11011. Springer, Cham.
https://doi.org/10.1007/978-3-319-98355-4_29

Branch, R. (2009). Instructional design: The ADDIE approach. Berlin, Germany: Springer-Verlag.

Corradini, I., Lodi, M., Nardelli, E. (2018). An Investigation of Italian Primary School Teachers' View on Coding and Programming. 11th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2018, Oct 2018, St. Petersburg, Russia. pp.228-243.
https://doi.org/10.1007/978-3-030-02750-6_18.hal-01913059

Desimone, L. M. (2009). Improving impact studies of teachers' professional development: Toward better conceptualizations and measures. Educational Researcher, 38, 181-199.
https://doi.org/10.3102/0013189X08331140

Dixson, M. (2015). Measuring student engagement in the online course: the online student engagement scale (OSE). Online Learning, 19(4), 1-15.
https://doi.org/10.24059/olj.v19i4.561

Eriksson, T., Adawi, T., & Stöhr, C. (2017). Time is the bottleneck: A qualitative study exploring why learners drop out of MOOCs. Journal of Computing in Higher Education, 29(1), 133-146.
https://doi.org/10.1007/s12528-016-9127-8

Ertmer, P. A., Ottenbreit-Leftwich, A. T., Sadik, O., Sendurur, E., & Sendurur, P. (2012). Teacher beliefs and technology integration practices: A critical relationship. Computers & Education, 59, 423-435.
https://doi.org/10.1016/j.compedu.2012.02.001

Google, Inc., & Gallup, Inc. (2016). Trends in the state of computer science in U.S. K-12 schools.
http://services.google.com/fh/files/misc/trends-in-the-state-of-computer-science-report.pdf

Grandell, L., Peltomäki, M., Back, R. J., & Salakoski. T. (2006). Why complicate things? Introducing programming in high school using Python. In Proceedings of the 8th Australasian Conference on Computing Education, 52, 71-80.

Gregori, E. B., Zhang, J., Galván-Fernández, C., & Fernández-Navarro, F. D. A. (2018). Learner support in MOOCs: Identifying variables linked to completion. Computers & Education, 122, 153-168.
https://doi.org/10.1016/j.compedu.2018.03.014

Gretter, S., & Yadav, A. (2016). Computational Thinking and Media & Information Literacy: An Integrated Approach to Teaching Twenty-First Century Skills. TechTrends, 60, 510-516.
https://doi.org/10.1007/s11528-016-0098-4

Guniš, J., Šnajder, L., Tkáčová, Z., & Gunišová, V. (2020). Inquiry-Based Python Programming at Secondary Schools. 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2020, 750-754.
https://doi.org/10.23919/MIPRO48935.2020.9245275

Ho, A., Chuang, I., Reich, J., Coleman, C., Whitehall, J., Northcutt, C., Williams, J., Hansen, J., Lopez, G., & Peterson, R. (2015). HarvardX and MITx: Two years of open online courses. Cambridge: HarvardX.

Hodges, C., Lowenthal, P., & Grant, M. (2016). Teacher professional development in the digital age: Design considerations for MOOCs for teachers. In Proceedings of Society for Information Technology & Teacher Education International Conference (pp. 2075-2081). Chesapeake, VA: Association for the Advancement of Computing in Education (AACE).

Huin, L., Bergheaud, Y., Caron, P. A., Codina, A. & Disson, E. (2016). Measuring completion and dropout in MOOCs: A learner-centered model, Proceedings of the European MOOC Stakeholder Summit 2016, 55-67.

ISTE. (2014). Operational definition of Computational Thinking in K-12 education.

Kellogg, S., & Edelmann, A. (2015). Massively open online course for educators (MOOC-Ed) network-dataset. British Journal of Educational Technology, 46(5), 977-983. https://doi.org/10.1111/bjet.12312

Kesim, M., & Altinpulluk, H. (2015). A Theoretical Analysis of Moocs Types from a Perspective of Learning Theories. Procedia - Social and Behavioral Sciences, 186(2015), 15-19. https://doi.org/10.1016/j.sbspro.2015.04.056

Klimeková, E., & Tomcsányiová, M. (2018). Case Study on the Process of Teachers Transitioning to Teaching Programming in Python. In: Pozdniakov S., Dagienė V. (eds) Informatics in Schools. Fundamentals of Computer Science and Software Engineering. ISSEP 2018. Lecture Notes in Computer Science, vol 11169. Springer, Cham. https://doi.org/10.1007/978-3-030-02750-6_17

Kostopoulos, G., Panagiotakopoulos, T., Kotsiantis, S., Pierrakeas, C., & Kameas, A. (2021). Interpretable Models for Early Prediction of Certification in MOOCs: A Case Study on a MOOC for Smart City Professionals, "Interpretable Models for Early Prediction of Certification in MOOCs: A Case Study on a MOOC for Smart City Professionals," in IEEE. https://doi.org/10.1109/ACCESS.2021.3134787

Krishnamurthi, S., & Fisler, K. (2019). Programming paradigms and beyond. In S. Fincher & A. Robins (Eds.), The Cambridge handbook of computing education research (pp. 377-413). Cambridge University Press. https://doi.org/10.1017/9781108654555.014

Kunkle, W. M. & Allen, R. B. (2016). The Impact of Different Teaching Approaches and Languages on Student Learning of Introductory Programming Concepts, ACM Transactions on Computing Education, January 2016 Article No. 3. https://doi.org/10.1145/2785807

Kwon, K. (2017). Novice programmer's misconception of programming reflected on problem-solving plans. International Journal of Computer Science Education in Schools, 1(4), 14-24. https://doi.org/10.21585/ijcses.v1i4.19

Lamprou, A., Repenning, A., & Escherle, N. (2017). The Solothurn project — Bringing computer science education to primary schools in Switzerland. In Proceedings of the 2017 ACM conference on innovation and technology in computer science education (ITiCSE 17), 218-223. New York: ACM.

Lazarinis, F., Karachristos, C.V., Stavropoulos, E.C., & Verykios, V. S. (2019). A blended learning course for playfully teaching programming concepts to school teachers. Education and information technologies, 24(2), 1237-1249. https://doi.org/10.1007/s10639-018-9823-2

Lloyd, M., Chandra, V. (2020). Teaching coding and computational thinking in primary classrooms: perceptions of Australian preservice teachers. Curriculum Perspectives, 40, 189-201. https://doi.org/10.1007/s41297-020-00117-1

Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. Computer Science Education, 16(3), 211-227. https://doi.org/10.1080/08993400600912384

Mason, S., & Rich, P. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. Contemporary Issues in Technology and Teacher Education, 19(4), 790-824. https://www.learntechlib.org/primary/p/184723

Mészárosová, E. (2015). Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools? International Journal of Information and Communication Technologies in Education, 4(2), 5-14. https://doi.org/10.1515/ijicte-2015-0005

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: a systematic review of the literature. Journal of Computers in Education, 5, 149-174. https://doi.org/10.1007/s40692-018-0101-5

Onah, F. O., Sinclair, J., & Boyatt, R. (2014). Dropout rates of massive open online courses: Behavioural patterns. In Proceedings of the 6th international conference on education and new learning technologies, Barcelona (EDULEARN14), 5825-5834. Spain.

Panagiotakopoulos, T., Kotsiantis, S., Borotis, S., Lazarinis, F., & Kameas A. (2021). Applying Machine Learning to Predict Whether Learners Will Start a MOOC After Initial Registration. In: Maglogiannis I., Macintyre J., Iliadis L. (eds) Artificial Intelligence Applications and Innovations. AIAI 2021 IFIP WG 12.5 International Workshops. AIAI 2021. IFIP Advances in Information and Communication Technology, vol 628. Springer, Cham. https://doi.org/10.1007/978-3-030-79157-5_38

Pérez-Foguet, A., Lazzarini, B., Giné, R., Velo, E., Boni, A., Sierra-Castañer, M., Zolezzi, G., & Trimingham, R., (2018). Promoting sustainable human development in engineering: Assessment of online courses within continuing professional development strategies. Journal of Cleaner Production, 172, 4286-4302. https://doi.org/10.1016/j.jclepro.2017.06.244

Poultsakis, S., Papadakis, S., Kalogiannakis, M., & Psycharis, S. (2021). The management of Digital Learning Objects of Natural Sciences and Digital Experiment Simulation Tools by teachers. Advances in Mobile Learning Educational Research, 1(2), 58-71. https://doi.org/10.25082/AMLER.2021.02.002

Rich, P. J., Browning, S. F., Perkins, M., Shoop, T., & Yoshikawa, E. (2018). Coding in K-8: International trends in teaching elementary/primary computing. TechTrends, 63, 311-329. https://doi.org/10.1007/s11528-018-0295-4

Robins, A. V. (2019). Novice programmers and introductory programming. In S. Fincher & A. Robins (Eds.), The Cambridge handbook of computing education research (pp. 327–376). Cambridge, University Press. https://doi.org/10.1017/9781108654555.013

Sands, P., Yadav, A., & Good, J. (2018). Computational Thinking in K-12: In-service Teacher Perceptions of Computational Thinking. In: Khine M. (eds) Computational Thinking in the STEM Disciplines. Springer, Cham. https://doi.org/10.1007/978-3-319-93566-9_8

Scherer, R., Siddiq, F., & Viveros, B. S. (2018). Technology and the mind: Does learning to code improve cognitive skills? In Proceedings of the Technology, Mind, & Society 2018 Conference. https://doi.org/10.1145/3183654.3183658

Scherer, R., Siddiq, F., & Viveros, B. S. (2018). Technology and the mind: Does learning to code improve cognitive skills? In Proceedings of the Technology, Mind, & Society 2018 Conference. https://doi.org/10.1145/3183654.3183658

Spradling, C., Linville, D., Rogers, M. P., & Clark, J. (2015). Are MOOCs an appropriate pedagogy for training K-12 teachers computer science concepts? Journal of Computer Science in Colleges, 30(5), 115-125.

Toikkanen, T., & Leinonen, T. (2017). The Code ABC MOOC: Experiences from a coding and computational thinking MOOC for Finnish primary school teachers. In P. J. Rich & C. B. Hodges (Eds.), Emerging research, practice, and policy on computational thinking (pp. 239–248). New York, NY: Springer. https://doi.org/10.1007/978-3-319-52691-1_15

Tuomi, P., Multisilta, J., Saarikoski, P., & Suominen, J. (2018). Coding skills as a success factor for a society. Education and Information Technologies, 23, 419-434. https://doi.org/10.1007/s10639-017-9611-4

Tzimopoulos, N., Provelengios, P., & Iosifidou, M. (2021). Implementation and evaluation of a remote seminar on the pedagogical use of educational robotics. Advances in Mobile Learning Educational Research, 1(2), 48-57. https://doi.org/10.25082/AMLER.2021.02.001

Vaca-Cárdenas, L. A., Bertacchini, F., Tavernise, A., Gabriele, L., Valenti, A., Olmedo, D. E., & Bilotta, E. (2015). Coding with Scratch: The design of an educational setting for Elementary pre-service teachers. 2015 international conference on Interactive Collaborative Learning (ICL), Florence, Italy (pp. 1171), IEEE.

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. ACM Transactions on Computing Education (TOCE), 18(1), 3. https://doi.org/10.1145/3089799

Yadav, A., Gretter, S., Good, J., & McLean T. (2017). Computational Thinking in Teacher Education. In: Rich P., Hodges C. (eds) Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations. Springer, Cham. https://doi.org/10.1007/978-3-319-52691-1_13