MEDIA EDUCATION
Studi, ricerche, buone pratiche

Firenze University Press
www.fupress.com/me

OPEN ACCESS

Citation: M. Di Stasio, B. Donati, M. Bianchini (2023) How teacher training on analytic philosophy can help schools in developing a decommodified interpretation of coding activities. *Media Education* 14(1): 41-47. doi: 10.36253/me-13466

Received: July, 2022

Accepted: July, 2022

Published: May, 2023

Copyright: © 2023 M. Di Stasio, B. Donati, M. Bianchini. This is an open access, peer-reviewed article published by Firenze University Press (http://www.fupress.com/me) and distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Competing Interests: The Author(s) declare(s) no conflict of interest.

How teacher training on analytic philosophy can help schools in developing a decommodified interpretation of coding activities[1]

Come insegnare la filosofia analitica ai docenti può aiutare la scuola a sviluppare un'interpretazione non mercificata delle attività di coding

Margherita Di Stasio[1,*], Beatrice Donati[2], Matteo Bianchini[3]

[1] *INDIRE, Florence, Italy*
[2] *University of Florence, Italy*
[3] *Scuola Città Pestalozzi, Italy*
m.distasio@indire.it; beatrice.donati@unifi.it; matteo.bianchini@pestalozzi.wikischool.it
*Corresponding author

Abstract. The term 'coding' is used to label a wide range of learning activities directly or indirectly related to the design and implementation of algorithmic procedures. These activities are now widespread in all levels of schools and are performed in very different pedagogical frameworks. Our hypothesis is that teachers look at coding from a perspective strongly influenced by their background. In particular, we deem that a logical and philosophical training may foster a de-commodified vision of coding and computational thinking. We have therefore decided to design and experiment a teacher training path that provides participants with some basics of logic and analytical philosophy. We analysed if and how this kind of training influences classroom coding activity. In the first part of this contribution (Sections 1 to 3) we introduce the theoretical framework underlying our hypothesis as well as the training and educational activities proposed to teachers. In the second part (Sections 4 to 7), we present the results of this path by means of a comparative case study focusing on the coding practices implemented in the experiences.

Keywords: coding, computational thinking, analytic philosophy, logic, teacher training.

Riassunto. Il termine 'coding' è utilizzato come etichetta per un novero di attività di apprendimento collegate, direttamente o indirettamente, al design e all'implementazione di processi algoritmici. Queste attività sono diffuse in ogni livello di scuola e portate avanti in contesti pedagogici molto diversi. La nostra ipotesi è che l'approccio al coding dei docenti sia fortemente influenzato dal loro quadro culturale. In particolare, riteniamo che una formazione logica e filosofica possa favorire una visione de-mercificata

[1] Although this contribution has been jointly conceived, Matteo Bianchini wrote sections 3.2, and 6.3; Margherita Di Stasio developed sections 1, 2.1, 3, 3.3, 4, 5, 5.1, 6.1 and 7; Beatrice Donati elaborated sections 2.2, 3.1 and 6.2.

della coding e del pensiero computazionale. Abbiamo quindi deciso di progettare e sperimentare un percorso di formazione per insegnanti che fornisca ai partecipanti alcune basi di logica e filosofia analitica. Abbiamo analizzato se e come questo tipo di formazione influenzi l'attività di coding in classe. Nella prima parte di questo contributo (Sezioni da 1 a 3) presentiamo il quadro teorico alla base della nostra ipotesi e le attività formative e didattiche proposte agli insegnanti. Nella seconda parte (Sezioni da 4 a 7), presentiamo i risultati di questo percorso come di studio comparato incentrato sulle pratiche di coding realizzate nelle esperienze.

Parole chiave: coding, pensiero computazionale, filosofia analitica, logica, formazione docenti.

1. INTRODUCTION[2]

At present, coding seems to be the mark of a school that wants to bring students to the future by bridging the supposed gap between formal instruction, job and technology. This perspective risks generating educational paths either aimed at providing a very premature job orientation or centred on the use of specific software or hardware. In these cases, "coding" is considered as the mere activity of "writing codes".

In an educational context, coding can also enhance rigorous reasoning, problem solving and deductive reasoning skills. It can give pupils useful tools for a deeper understanding of a world where the Internet of things, artificial intelligence and *infosphere* (Floridi, 2014) are basic components of the *onlife* reality (Floridi, 2015).

We will try instead to convey a different view of coding, which advocates a computational thinking not merely aimed at programming.

Our proposal presents analytic philosophy and logic as the most natural framework in which to develop coding skills and computational thinking, so as to acquire awareness about technologies.

In the first part of this contribution, we will present a framework of coding and computational thinking, as well as their introduction in school. We will also underline their relationship with logic and analytic philosophy.

In the second part, we will introduce our project based on this framework in terms of hypothesis and path proposal to the involved teachers.

We will then analyse the first results of data analysis.

2. CODING AND COMPUTATIONAL THINKING: A FRAMEWORK

2.1 Introducing coding and computational thinking in school

Inter omnes constat that Papert's expression 'computational thinking' (Papert, 1980) became viral thanks to a Jannette Wing's article (Wing, 2006). In this work, the author claims that «Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction» (Wing, 2006, p. 35). In the past decade, coding entered in national school curricula and in the academic debate.

Considering a school context, we can face the introduction of computational thinking and coding aimed at pupils and teachers.

In *Computational Thinking. A Guide for Teachers*, Csizmadia et al. (2006) defined computational thinking as a «process of recognising aspects of computation in the world that surrounds us and applying tools and techniques from computing to understand and reason about natural, social and artificial systems and processes» (Csizmadia et al., 2006, p. 5).

With regard to students' competence and curricula, in *Indicazioni nazionali e nuovi scenari* (MIUR, 2018), the Italian Ministry of Education introduced coding in curricula as a path for education to logical and analytical thinking that can be a useful ground to develop mathematical, scientific and technological skills, as well as language skills refinement.

In the same year, from a broader perspective, *A Global Framework of Reference on Digital Literacy Skills for Indicator 4.4.2* (Law et al., 2018) underlines that the current focus of computational thinking is on algorithmic thinking as an integral part of problem-solving competences in the digital world and it does not necessarily involve programming in specific computer languages.

From this point of view, coding and computational thinking can be significant competences in a broad context of civic engagement and education. A recent approach looks at competencies as built on Core Foundations that include a digital dimension: «core knowledge, skills, attitudes and values for 2030 will cover not only literacy and numeracy, but also data and digital literacy, physical and mental health, and social and emotional skills» (OECD, 2019).

We can also observe an increasing interest in teacher training: neither coding nor computational thinking had been considered in UNESCO ICT Competency

[2] The findings presented are part of the research funded by the project "Didattica Laboratoriale Mulitidisciplinare" (10.8.4.A2-FSEPON-INDIRE-2017-1) of the National Operational Program (PON).

Framework for Teachers in 2011, but they were both introduced in 2016.

With regard to teachers, some recent systematic reviews analyse the impact of coding introduction in teaching practices. One of these reviews shows a lack of project support to teachers for them to orientate coding activities to the achievement of skills (Frison, 2019). Another one furthermore underlines that few studies have been published on the training of primary school teachers and that training is more based on basic computer skills than on pedagogical aspects (Mason & Rich, 2019).

2.2 Relationship between coding and logic as a discipline

Logic has a long and recognizable history and its collection of methods, notions and theories make it a well-defined discipline in the academic context (Gabbay et al., 2004; Cantini & Minari, 2019). However, as soon as we move into the world of pre-academic school projects, "logic" as a discipline disappears. Although great importance is attached to "logical skills" and "deductive reasoning", logic is never taught as a subject in itself. Formal logic is in fact the big absentee of our schools. From our viewpoint, this absence is primarily the evidence of a perspective error, whereas coding is seen as a practical tool, immediately usable in the job market.

To see things the other way round, if we bring computer programming back to the theoretical framework which it belongs to, we can clearly see why formal logic should not only be present, but also be the very first step of the computational thinking training of teachers and pupils. As a matter of fact, the history of science helps us understand the relationship between logic and coding, as the first programming languages were actually created as a consequence of the studies of formal logic. It is theoretical computer science that follows from formal logic, rather than vice versa (Davis, 2000).

Formal logic is related to coding for different reasons. The first one is related to the idea of 'programming language' itself. Programming languages are in fact a special kind of formal language, specifically designed to express procedures and algorithms (del Vado Virseda, 2019). For this reason, we decided to give teachers some general notions of formal languages, starting from the very basic notion of 'logical formalization' and discussing the relationship between formal and natural languages. The second link between logic and coding is less direct but just as interesting. If we present coding as a way of enhancing computational thinking, we are certainly interested in stimulating correct reasoning in our pupils. However, it is not easy to do it without a clear and explicit notion of what correct reasoning is. Logic helps us also in this context, by providing a formal definition of deduction and correctness.

3. BRIDGING PHILOSOPHY AND CODING

Starting from the framework described above, we want to highlight the role of computational thinking in teacher training as the basis of the coding activity, as well as the relationship between code and languages, based on formalization processes.

We propose analytic philosophy and logic as the theoretical ground to approach this perspective. Our hypothesis is that a training specifically focusing on logic and philosophy of language can help teachers build awareness about the deepest aspects of coding and computational thinking. In particular, we identified a path from Leibniz to Frege and Wittgenstein which underlies the relationship among natural languages, formalization and programming languages. In this perspective, coding is a way to understand our world, to read and write the technological reality.

Our research questions are therefore as follows:

- can a training path based on analytic philosophy influence the teachers' stance on coding and change their way to design coding activities?

And can it, consequently, foster computational thinking skills?

- can this kind of path help to acquire awareness about technologies?

3.1 Topics of teacher training in logic

When choosing the specific topics for our logic training, we selected notions that do not require any specific philosophical or mathematical background but that, at the same time, could provide a self-contained set of notions useful in different school contexts. The general topics we wanted to address, as already clarified in the previous sections, where 'formal languages' and 'algorithms'. To do that, we started with the introduction of the fundamental notion of 'logic formalization', by choosing in particular the rules of Classical Logic. This choice is not arbitrary, since it is widely accepted as the more natural starting point of any formal analysis of our everyday reasoning, even with its well-known limits. It is then possible to introduce a formal definition of *deductions* and *logical arguments* based on the same logical rules. Each proposed topic was accompanied by a rich set of examples and exercises. Finally, since the training was devoted to coding activities, we proposed a focus on *procedural logic* and *algorithms*. Here follows the complete list of topics.

The first part is devoted to classical propositional logic:

- From natural to formal language: the formalization process;
- The language of "classical propositional logic";
- Truth tables;
- The limits of classical logic.

The second part focuses on the concept of deduction:

- Deductive arguments;
- Notable arguments and fallacies;
- Quantifiers and first order logic;
- Aristotelian syllogism.

The third part focuses on some peculiar aspects of coding:

- Algorithms;
- Formal and pseudo-formal languages;
- Basic syntax of procedural languages (variables, flow controls, boolean operators);
- Flow Charts.

3.2 Bridging training on philosophy and logic and coding activity

To design learning activities focusing on the cognitive processes involved in computational thinking and in the approach to coding skills, we intended to bridge teachers training to coding activity.

We propose to teachers a simulation activity divided into two successive phases to achieve the construction of learning.

The first phase, consisting of two activities, aims at coding capacity: trainees must recognise the need for a formal language that cuts down as much as possible the ambiguities of communication.

The first activity is designed to allow trainees to recognize the need for a formal language that cancels the ambiguities of communication, the second one shifts from communication to formulation of a command.

In the second phase, it is possible to restrict the fields of communication and command by means of activities that avail themselves of robots and programming environments. Thus, the logical and algorithmic language may be used to approach formal programming languages.

3.3 Proposal to design learning experience

For the declination of the learning activity, we chose the methodology of Problem Based Learning (PBL) for some peculiar characteristics highlighted by different perspectives, which we considered suitable for the project.

According to Nilson (2010), PBL sustains the possibility of working in teams as well as independently, promoting critical thinking, analysis, explanation of concepts, in particular across disciplines.

According to Astolfi (1993) the problem-situation works as a scientific debate within the classroom and motivation to build the solution supports students to develop or find together the appropriate intellectual and operational tools.

We have proposed to the teachers a model to design and realise the learning activities (Maccario 2010; Tessaro, 2014) articulated in the following phases: presentation to the class of an objective/obstacle that: takes into account students' foreknowledges, asks the involvement of more disciplines in order to reach a solution, presents a manageable complexity, but such as to require a modelling; study and analysis; formulation of the hypothesis, research and experimentation; plenary discussion; synthesis and generalisation.

4. RESEARCH METHODS

The project was built up on a design-based model.

Design-Based Research (DBR) is a methodology that particularly focuses on supporting the innovation of educational processes; it is explicitly aimed at producing a change, thanks to a collaboration between researchers and practitioners (Design-Based Research Collective, 2003).

Data collection and outcome analysis are carried out in the form of a comparative case study (Merriam, 1998).

The case study is a method used in training contexts and in educational research. It is used in training as a tool of reflection in self-training and peer-to-peer training (Calvani et al. 2007). In this perspective, the case study can be proposed in two ways: a) by analyzing a ready-made case, either real or simulated; b) by reconstructing a real case or by constructing a plausible one while analyzing variants and formulating hypotheses.

In educational research, case study is a questioned methodology. It is used in action research, sometimes in contexts which are not consistent with this tool, and this is why its rigour has been questioned. The case study makes it possible to investigate a phenomenon within its real context, especially when the boundaries between such phenomenon and the context are not well defined (Yin, 2009). The object must be circumscribed, limited in the observation time and in the subjects who take part in it. This allows to achieve a final product, focused on a particular situation; conspicuously descriptive of the phenomenon examined; heuristic, clarifying the understanding of this phenomenon (Merriam, 1998).

In this case, we have a narrow framework, since the project involves the classes of two schools whose coding experiences are analysed here. We intend to describe the resulting experiences in a broad and in-depth way to understand if the proposed path has been effective and what elements can be used to model the experience.

This tool is therefore consistent with our research.

To have the descriptive wealth typical of the case study methodology, data can be collected with many methods (Merriam, 1998) and in this experience we have used:

- an *on desk* documentary analysis of lesson plans, diaries, p2p observations, photos and videos of school activities;
- data collection *in field* (to be performed online due to Covid-19 restrictions): interviews with teachers and focus groups with students.

5. THE CONTEXT

We started with experiential research involving 2 Omnicomprensivo, namely, school institutions ranging from primary to secondary and to high school. They participated engaging 13 classes. We involved teachers who usually taught language-related subjects: grammar and coding in primary and secondary school and Italian grammar, Literature and History of philosophy in high school.

The first step was a seminar, aimed at presenting the path to the involved teachers and at kicking off the first part, devoted to training on theoretical elements with lessons and talks with experts. We provided teachers with theoretical material and tools to design teaching and learning activities, to document and reflect on them.

We analysed, discussed lesson plans and provided teacher orientation during the revision process of their proposals. In spite of the restrictions due to the pandemic emergency, we managed to observe the results of school activities and to discuss them with teachers and pupils.

In this contribution, we will analyse the coding activities carried out.

5.1 Teachers: basic features and choices

With regard to the background of the teachers involved, 1 teacher out of 3 declares that they have a former education in philosophy; 3 teachers out of 3 do not have any former education in coding; 3 teachers out of 3 have never experienced coding activities. If compared to the usual teaching activities, 2 are teachers of mathematics and science, 1 of humanities.

In one case – we will indicate it as an 'unplugged experience' – we have 2 teachers of different subjects who collaborate in the same class V of 10 pupils of a small mountain centre. They choose to design a coding path together: the 12-hour path is centred on unplugged activities and paper/pencil; they tend not to envisage the use of computers or robotic mediators if not stressed.

In the other case – we will indicate it as 'plugged experience' – the teacher of mathematics and science who designs by themselves the coding experiences in a class IV composed of 18 students in a regional capital city. The 10-hour path provides an in-depth study of computer science as well as coding sessions.

6. DATA COLLECTION AND ANALYSIS

6.1 Themes and focus

While *on desk* documentary analysis, we can detect similarities and differences between the plugged and the unplugged experience in relation to the themes chosen in the design phase – from lesson plans – and carried out in the development of experience – from diaries.

Table 1. Context, sample and involvement in coding experiences.

	Istituto Onnicomprensivo *Convitto Nazionale Colombo* Genova		Istituto Onnicomprensivo *San Marcello Pistoiese* Pistoia	
Primary	2 teachers	3 classes	2 teachers	1 class
Secondary	2 teachers	1 class, 1 extra-curricular class	1 teacher	1 class
High School	2 teachers	2 classes	2 teachers	2 classes
Involvement in coding experiences	1 primary teacher of mathematics and science 1 class IV of 18 pupils		2 primary teachers of different subjects (mathematics and science, humanities) 1 class V of 10 pupils	

Which thematic nodes did teachers choose during the project stage?

The themes emerging from lessons plan can be summarised as follows: relationship between natural language and formal language; understanding of natural language; possibility of formalisation; detection of connectives and logic elements; flowcharts; algorithms. From lesson plans, both proposals are centred on the algorithmic part and lesson plan on 'plugged experience' also pays special attention to the theme of natural language, its analysis and the relationship between natural and formalised languages.

What was the focus of the experiences?

Also in teaching practices, as emerge from diaries, both experiences are especially focused on algorithms.

In addition, the unplugged experience focuses on proceeding, symbol, understanding, while the plugged one on clarity/ambiguity; sequence; machine.

6.2 Positive aspects and critical issues

This first experience allowed us to obtain interesting results during classroom experimentation and at the same time highlighted some criticalities of our proposal. With regard to the positive results, relevant aspects emerge from *on desk* analysis of lesson plans and diaries as well as from *in field* revelations through interviews. Teachers involved in the project gave particular importance to unplugged activities in coding classes. This should be seen as a focus shift from machine programming to abstract reasoning, that was exactly our main interest.

For similar reasons, we find very significant that a discussion about the relationship between formal and natural language has been proposed in every class. This represents an interesting novelty, because it adds some elements of philosophy of language to a coding activity.

Another positive aspect is that, during the process, the teachers specifically requested an in-depth study of algorithms and flow diagrams. This proactive attitude denotes the interest of participants.

On the other hand, we have identified some weaknesses in our proposal. In most cases, from on desk analysis as well as from interviews, emerges that teachers turned out to be hardly able to deepen logic in a self-training mode. This should not be thoroughly surprising, since Logic is a really formal subject and the participants should be probably provided with a more structured training path, with synchronous lessons, group activities and guided exercises.

6.3 Educational possibility

Coding encourages non-directive learning by discovery, thanks to heuristic solutions.

We believe it is important to pay special attention to the analysis and use of errors in training, particularly in the initial phase. In fact, errors are experienced in many didactic ideas as frustrating or, in any case, as having a negative value. In coding, instead, errors are positively interpreted as a chance to trigger analysis.

Pupils can work in an environment aimed not only at programming, but also at analysing logic nodes. In this path, pupils autonomously produce and test their own ideas.

This activity, based on philosophy of language and logic, seems to enhance a critical approach to thinking skills.

7. CONCLUSIONS

Having analysed positive aspects and critical issues, as to our hypothesis and research questions, we can draw some conclusions from our analysis, in particular about the relationship about technological awareness.

Is There a Change in the Perception of Technologies?

The focus group created with pupils brought out the perception of coding as a "lingua franca", that enables human-machine interaction. For instance, one child says: «You give instructions through a code. We didn't use natural language, we used an algorithm».

What Kind of Approach to Have with Technology?

As Wing maintains, «thinking like a computer scientist means more than being able to program a computer» (Wing, 2006). We believe that giving teachers a philosophical perspective can help them perceive the cultural aim of coding activities. Therefore, this kind of training seems to help teachers approach coding as a reflective practice to sustain processes of analysis and modelling rather than as a drill activity.

What Kind of Interaction with our World?

As the Onlife Manifesto (Floridi, 2015) says:

«ICTs are not mere tools but rather environmental forces that are increasingly affecting:

1. our self-conception (who we are);
2. our mutual interactions (how we socialise);
3. our conception of reality (our metaphysics); and
4. our interactions with reality (our agency).

In each case, ICTs have a huge ethical, legal and political significance».

A preliminary analysis shows that this path can bring pupils to look at coding as a way to interact with the technological elements of their ordinary world.

We think that this will pave the way for pupils to become an active part of our world and society.

REFERENCES

Astolfi, J.P. (1993). Placer les élèves en "situation-problème"?. *Probio-Revue*, vol. 16, no 4.

Calvani, A., Ranieri, M. & Bonaiuti, G. (2007). *Fondamenti di didattica: teorie e prassi dei dispositivi formativi*. Carocci.

Cantini, A., & Minari, P. (2019). *Introduzione alla logica: Linguaggio, significato, argomentazione*. Le Monnier.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computational thinking. A guide for teachers*. Retrieved from: https://eprints.soton.ac.uk/424545/1/150818_Computational_Thinking_1_.pdf.

Davis, M. (2000). *The Universal Computer: The Road from Leibniz to Turing*. W. W. Norton and Company.

del Vado Vírseda, R. (2019). Computability and Algorithmic Complexity Questions in Secondary Education. *Proceedings of the ACM Conference on Global Computing Education (CompEd '19)*, Association for Computing Machinery, pp. 51–57.

Design-Based Research Collective (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, 32(1), 5-8.

Floridi, L. (2015). *The Onlife Manifesto: Being Uuman in a Hyperconnected Era*. Springer Open.

Floridi, L. (2014). *The fourth revolution: How the infosphere is reshaping human reality*. Oxford OUP.

Frison, D. (2019). Educational robotics in the early childhood settings 0-6: a systematic review. *Form@re - Open Journal Per La Formazione in Rete*, 19(1), 30-46.

Gabbay, D. M., Woods, J., & Kanamori, A. (2004). *Handbook of the history of logic*. Elsevier.

Law, N., Woo, D., de la Torre, J., & Wong, G. (2018). *A global framework of reference on digital literacy skills for indicator 4.4. 2*. Unesco Institute for Statistics.

Maccario, D. (2010). *A scuola di competenze*. SEI.

Mason, S. L. & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education*, 19(4), 790-824.

Merriam, S.B. (1998). *Qualitative Research and Case Study Applications in Education. Revised and Expanded from "Case Study Research in Education"*. Jossey-Bass.

MIUR (2018). *Indicazioni nazionali e nuovi scenari*. Retrieved form: https://www.miur.gov.it/documents/20182/0/Indicazioni+nazionali+e+nuovi+scenari/3234ab16-1f1d-4f34-99a3-319d892a40f2.

Nilson, L. B. (2010). *Teaching at its best: A research-based resource for college instructors*. Jossey-Bass.

OECD (2019). *OECD Learning Compass 2030 - Concept Note Series*. Retrieved from: https://www.oecd.org/education/2030-project/contact/OECD_Learning_Compass_2030_Concept_Note_Series.pdf.

Papert, S. (1980). *Mindstorms*. Basic Book.

Tessaro, F. (2014). Compiti autentici o prove di realtà?. *FORMAZIONE & INSEGNAMENTO. Rivista internazionale di Scienze dell'educazione e della formazione*, 12(3), 77-88.

Wing, J. M. (2006). Computational Thinking. *Communication of ACM*, 49(3), 33-35.

Yin, R. K. (2009). *Case study research: design and methods*. SAGE Publications.