CODING FOR INCLUSION

www.codinc.fun

Co-funded by the Erasmus+ Programme of the European Union

METHODOLOGY C0D1NC

INTRODUCTION

This methodology addresses teachers in secondary and primary schools or animators.

This methodology will explain to the user how to engage students through a stimulating pedagogical methodology that will have students from secondary schools (aged 15 and over) to teach basic coding and STEAM education to their younger peers - pupils aged 8-12 years.

During the CODINC project, we train the students and their teachers in the basics of coding. Our trainers will provide 10 hours of coding education in the classroom.

After, the training, the secondary school students will coach pupils in primary schools accompanied by their teachers and our animators.

For the teachers in secondary schools, conducting this project will help them in raising their science capital, develop a pedagogical attitude and coaching skills and to transmit the skills of basic coding to the pupils aged from 8 till 12.

The CODINC project focuses on promoting inclusion through a peer to peer STEAM and coding educational training programme. The students and pupils targeted in the CODINC project are specifically from neighbourhoods where there may be more exclusion, this is primarily measured through measured disadvantaged in comparison to other areas. Many country offer an index identifying schools or areas with higher levels of exclusion and disadvantages. The CODINC project helps raise the confidence of students aged 15-18 from disadvantaged backgrounds by showing helping them discover their creative, innovative and supportive potential. In practice this is done by training the students in a programme where they learn how to programme, create their own coding and STEAM education projects, and then go to teach students. This helps students who may be struggling with identity and are looking to find a place in an adult world. Students of disadvantaged backgrounds may have the feeling to be considered as outsiders, with the trust and confidence from adults to discover their creative and innovative potential.

CODINC will adapt, disseminate and scale up the inclusive learning “Capital Digital” practice, which was developed and implemented by the project partner Maks vzw in neighbourhoods in Brussels, Belgium that are classified as highly disadvantaged.

Capital Digital is a programme that employs an innovative peer-to-peer learning methodology by training young people from disadvantaged backgrounds in STEAM and coding to become animators to their younger peers. In practice Capital Digital successfully trained 15-18-year-old students from disadvantaged backgrounds to teach coding and programming to their younger 10-12-year-old peers. The young "animators" learned to engage pupils in STEAM and coding activities in a playful way. This programme became the first work experience for the young animators which enhanced their confidence, which in turn broadened their horizons when deciding on a fitting educational and career path. The programme allowed them to connect with each other in a constructive way and to enjoy the role of educating their younger peers. Moreover, the project effectively supported young people by developing their critical thinking, creativity, digital and collaborative skills, and science capital. Most importantly, the Capital Digital peer-learning pedagogical method has a strong inclusive potential and fosters the STEAM education and the inclusion of disadvantaged students both inside and outside the classroom.

The CODINC project will adapt and scale up the Capital Digital methodology (and training Toolkit) into a different educational context (schools - formal Education) and pilot it in 5 European countries, namely Belgium, Cyprus, Germany, Italy and Spain. The specific objectives of CODINC are the following:

1. Increase and improve teachers' capacity to foster the STEAM education of disadvantaged youth in an inclusive educational approach based on peer learning
2. Empower disadvantaged young people in the acquisition and development of IT and collaborative competences as well as problem solving, self-confidence and creativity through a peer-learning training programme on Coding
3. Foster the development of a European "Coding for Inclusion" learning community among different actors and across different sectors (formal and non-formal education and training) able to sustain project results and amplify their impact.

There is a need across Europe to develop the digital competences of all citizens, in many countries the educational system is still not fully adapted to the use of ever-new technologies. In neighbourhoods where there are higher rates of exclusion, there is even more of a need to use digital tools to promote inclusive practices While pupils and students are active internet and mobile device users, teachers are needing to develop their skills to adequately support students in new technologies and competences. STEAM technology and coding offer new pathways for teaching, based on the interest of kids and students, enabling them to be producers, and not just consumers, of digital content.

CODINC will directly address:

- Primary school pupils (8 to 12 years of age) and secondary school students (15 to 18 years of age) from disadvantaged areas. We name the first group pupils and the second group students.
- Primary and secondary school teachers

- Trainers working with young people in formal, non-formal and informal settings (schools, telecentres, youth centres, NGOs, training centres etc.)
- Key stakeholders: schools, families, civil society organisations, local communities, NGOs, educational institutions and learning providers, public authorities, policy makers, etc.

In this methodology, we will describe a training schedule for the implementation for a 10-hour training course of coding and pedagogical skills for the students and a training schedule for the students and teachers. There are many similarities between the program proposed to the students and the pupils. This is done so because the students should be trained on the same modules they will use in turn to teach the pupils. The training program for students and teachers focusses also on pedagogical methodology of coding, evaluation techniques and energizers. Through training the students easy to grasp concepts, their self-confidence to teach and believe in their abilities along with their self-esteem will grow and the fear that they have to coach something they don’t really understand will disappear.

I hope you enjoy the reading!

LEARNING OUTCOMES

Using this methodology, your students and yourself will complete a wide range of learning tasks which will developing everything from computational thinking to creative and social skills.

The learning outcomes in the CODINC methodology are divided into two sections, one for primary and one for secondary education. Each section contains learning outcomes for the pupils (8 till 12) or students (+15) participating in the program and separate goals for you as a teacher.

You can consult the training goal matrices for a quick reference on what activity helps complete each learning goal.

Learning Outcomes for Teachers in Secondary Education and Primary Schools

As a teacher, you will firstly, gain valuable insights into computational thinking and coding and secondary strengthen your pedagogical skills.

After completing this methodology, you will have a better understanding of:

- What Computational Thinking is and why it matters in society?
- What are key 21st Century Competencies for your pupils?
- How coding activities foster computational thinking, creativity and problem-solving skills in pupils?
- What Pedagogical and Social Skills are essential for students to lead coding activities with their younger peers?

As an educational professional, you will learn to apply, guide and mentor training activities in

- Digital coding
- Offline computer science
- Collaborative and creative exercises
- Learning through design
- Pedagogical and animation activities.

The teachers of the secondary school are trained together with their students; the teachers of the primary school have a separate training.

Learning Outcomes for students

The students participating in the program will complete a large variety of learning outcomes, within the fields of:

- computational thinking
- collaborative and social skills
- creative skills
- problem solving and design skills
- pedagogical skills and insights into group animation.

These are of course not the only outcomes you can attain with this methodology, as your students will also be using verbal and visual languages in creative ways. You will even be applying mathematics and can decide to connect the final project to an educational subject of choice, such as history, geography, art and so on.

Computational Thinking

The learning outcomes for computational thinking are divided into two sections: the concepts that are to be mastered by the students and their corresponding cognitive practices.

There are nine key **concepts** for computational thinking in this methodology, with which students can successfully master coding and create their own games or programs.

Concept	Description
Algorithms	A written set of instructions for a computational device; for example, for a game, an app or even a cooking recipe.
Sequences	Instructions that are given in a discrete order, one following the other. The computer executes them from top to bottom.
Repetition and Loops	Repeating a subset of instructions several times (or infinitely) is called a loop.
Events and Selection	An indication of when an event should take place; for example, the cat starts moving when the 'start' button in a game is pressed.
Conditionals and Logical Operators	Letting the computer make a decision. If something happens, then an event should take place, or not it should not take place.
Mathematical Operators	Algorithms that require mathematics, such as multiplication, addition, … for example for reducing the speed of a ball over time.
Variables and Data Management	Variables are boxes in which numbers (or texts) can be stored. These boxes are then used in code, for example to show the current score of the player.

Functions	Reusing a subset of code, for example for walking forward, with a name label.

Consequently, there are six crucial **thinking practices**, that contribute to becoming a proficient coder and computational thinker. These thinking practices aren't just helpful in the field of technology, but transfer tremendously well to other fields of thinking and life, such as creative problem solving.

Thinking Practice	Description
Incremental and Iterative Work Strategies	Students break up their work into small steps, as well as return to previous steps in their thinking to improve their program. Coding is not a linear process, but a spiral that keeps repeating itself.
Testing and Debugging	Students can test a game or project against their expectations and learn lessons from it for improvement. They can identify problems (bugs) and use strategies to fix them (debugging).
Reusing and Remixing	Students can learn from projects made by others; reuse pieces of code or thought processes; and remix existing solutions or projects into something new.
Abstraction	Students can transfer the lessons learnt from a project into abstract patterns.
Modularization	Students can break projects down into smaller parts (i.e. movement, speed, score …) and reuse those parts in their work.
Information Collection and Management	Students can identify sources for information and look for solutions for their problems in various places, such as with peers or online.

Collaborative and Social Skills

The CODINC methodology explicitly chooses for a collaborative, peer-to-peer approach, in which older students (secondary education) work together to teach coding to younger pupils (primary education).

Your students will work on their collaboration skills, as they will be (1) working as a team and learning together. They will learn to (2) negotiate with each other, as well as with the pupils they will work with. Teaching activities require not only a knowledge of the subject, but also provides great pedagogical insights into the thinking of younger peers. Subsequently, they will learn to (3) describe their own thought processes, in a language that is appropriate for their target audience (whether that is the group of younger pupils, or their own age peers). Finally, they will learn from each other and from the younger group, through (4) vicarious experience and collaborative problem solving. Every student is a coach for 5 pupils. The students are taught not to give a course but to guide the pupils in discovering and solving problems. The role of the student of teacher is to accompany the pupils to find solutions to the given problems. The base are the working sheets: the student explains to task to the pupils and give them tips or tricks to resolve the problems.

A concrete example is the Scratch project. In th start of this project pupils discover the technical aspects of Scratch. We ask them to choose a figure and to make this figure moving around, dancing, saying of singing some things. They discover the bases of programming with Scratch. In the second part of the Scratch course, we ask them to do some exercises based on the working sheets. In the third and final phase, learners are asked to design a small game. This permits the learners not only to design a game but also think on the content of a game. When you design a game, you have to think about winning and losing, the different characters in your game. It's a kind of analysis from a problem that you have make.

The same methodology is used by students and pupils, so the students can experience how they will work with the pupils and this will help them understand how to best facilitate and engage the pupils in a learning process.

In the 10 h course, we talk also about the need of positive affirmation: give compliments every time you can. If you need to punish a child, it is better to isolate the situation and explain to him or her very clearly why you are doing this – punishing is needed only when a child break the houserules several times after a few warnings. It is better to not wait, but to do so immediately, so the child understands.

Creative Skills

Throughout the trajectory, creative skills are actively fostered. The three main aspects of creativity (according to the LEGO systemic creativity framework) are put to work: (1) exploring possibilities, ideas and other people's projects; (2) combining existing skills, ideas, projects and more into new creative endeavours; and (3) transforming existing properties into something fresh.

Problem Solving and Design Skills

Finally, the methodology also addresses problem solving and design skills.

Students will learn how to (1) identify a new problem and its components. They will also acquire strategies for (2) generating new ideas through brainstorming and (3) how to implement these ideas. Of course, they will then (4) evaluate their solution and reflect on the impact of it. Finally, they will improve on their solution and re-evaluate the problem in a next (5) iteration of the process.

Learning by trying things out is the base of this project and the way to reinforce computational thinking. For example with makey-makey, the bases are explained and then the pupils can start to try out and play with it.

Pedagogical and Animation Goals

In this program, students will acquire a great amount of pedagogical skills and insights into group animation. They will learn these both in the ten-hour-long training program, but even more by working actively with a group of pupils.

They will acquire knowledge and skills concerning (1) conducting group animation and activities; (2) managing individual and group motivation in pupils and using energizers to boost the morale; (3) evaluating activities, processes and their own performance as an animator; (4) setting group rules of conduct; and (5) finally practical organisation and insights.

The basic rules of conduct are developed in the start of the project with the pupils. We take the time to discuss with them this rules. We start with a big sheet of paper and try to formulate do's and don't's. If everyone agrees, all participants (pupils, students and teachers) sign this paper. We ask them to emphasize on positive formulation. Mostly this kind of rules is coming out of the discussion.

1. Treat others as you would like to be treated.
2. Respect other people's property and person and the materials used
3. Laugh *with* anyone, but laugh *at* no one.
4. Be responsible for your own learning.
5. Come on time.
6. Listen to the coach and to your friends to find solutions
7. Do not disturb people who are working.
8. Talk a language that everyone understands

The students are trained to have this discussion with the pupils. The same discussion is also organized with them in the start of the project.

In a class situation, it is a little bit different. It is important to discuss with the teachers how they want to deal with the rules of conduct and if there is no conflict with the rules of conduct of the school. We highlight the importance of letting the students work this out with the pupils.

This also means that the students are in charge of the authority in the classroom and wellbeing of the pupils during this project and that the primary school teachers are told to intervene only in case of big problems.

Learning Goal Matrix for Primary Education Pupils

The different activities of the CODINC methodology will of course address a varying selection of these learning goals. Throughout the program, all of the learning goals will be met. You can find a convenient overview of all the different learning goals -and what activities they are acquired in- in the **matrix for secondary education**.

Cfr annex learning goals matrix.

Primary Education Learning Goals

Learning Goals for Pupils (< 12 years old)

The pupils participating in the program will complete a large variety of learning goals, within the fields of (1) computational thinking; (2) collaborative and social skills; (3) creative skills; (4) and problem solving and design skills.

These are of course not the only goals you can attain with this methodology, as your pupils will also be using verbal and visual languages in creative ways. You will even be applying mathematics and can decide to connect the final project to an educational subject of choice, such as history, geography, art and so on.

Computational thinking

The learning goals for computational thinking are divided into two sections: the concepts that are to be mastered by the pupils and the corresponding thinking practices.

There are nine key concepts for computational thinking in this methodology, with which pupils can successfully master coding and create their own games or programs.

Concept	Description
Algorithms	A written set of instructions for a computational device; for example, for a game, an app or even a cooking recipe.
Sequences	Instructions that are given in a discrete order, one following the other. The computer executes them from top to bottom.
Repetition and Loops	Repeating a subset of instructions several times (or infinitely) is called a loop.
Events and Selection	An indication of when an event should take place; for example, the cat starts moving when the 'start' button in a game is pressed.
Conditionals and Logical Operators	Letting the computer make a decision. If something happens, then an event should take place, or not it should not take place.
Mathematical Operators	Algorithms that require mathematics, such as multiplication, addition, … for example for reducing the speed of a ball over time.
Variables and Data Management	Variables are boxes in which numbers (or texts) can be stored. These boxes are then used in code, for example to show the current score of the player.
Functions	Reusing a subset of code, for example for walking forward, with a name label.

Consequently, there are six crucial **thinking practices**, that contribute to becoming a proficient coder and computational thinker. These thinking practices aren't just helpful in the field of technology, but transfer tremendously well to other fields of thinking and life, such as creative problem solving.

Thinking Practice	Description
Incremental and Iterative Work Strategies	Pupils break up their work into small steps, as well as return to previous steps in their thinking to improve their program. Coding is not a linear process, but a spiral that keeps repeating itself.
Testing and Debugging	Pupils can test a game or project against their expectations and learn lessons from it for improvement. They can identify problems (bugs) and use strategies to fix them (debugging).
Reusing and Remixing	Pupils can learn from projects made by others; reuse pieces of code or thought processes; and remix existing solutions or projects into something new.
Abstraction	Pupils can transfer the lessons learnt from a project into abstract patterns.
Modularization	Pupils can break projects down into smaller parts (i.e. movement, speed, score …) and reuse those parts in their work.
Information Collection and Management	Pupils can identify sources for information and look for solutions for their problems in various places, such as with peers or online.

Collaborative and Social Skills

The CODINC methodology explicitly chooses for a collaborative approach, in which pupils explore, experiment and create a final project together. Because of this, a great variety of social and collaborative skills are acquired and practiced.

The pupils learn to (1) work together on a game in the final project, which teaches them to (2) negotiate with each other, as well as (3) describing their own thought processes.

Next to that, they also learn to (4) present their own projects in front of the groups, while (5) giving and receiving mutual feedback. Finally, they learn not only on their own, but even more from (6) vicarious, shared experiences.

Creative Skills

Throughout the trajectory, creative skills are actively fostered. The three main aspects of creativity (according to the LEGO systemic creativity framework) are put to work: (1) exploring possibilities, ideas and other people's projects; (2) combining existing skills, ideas, projects… into new creative endeavours; and (3) transforming existing properties into something fresh.

Problem Solving and Design Skills

Finally, the methodology also addresses problem solving and design skills.

The pupils will learn how to (1) identify a new problem and its components. They will also acquire strategies for (2) generating new ideas through brainstorming and how to (3) implement these ideas. Of course, they will then (4) evaluate their solution and reflect on the impact of it. Finally, they will improve on their solution and re-evaluate the problem in a next (5) iteration of the process.

Learning Goal Matrix for Primary Education Matrix

The different activities of the CODINC methodology will of course address a varying selection of these learning goals. Throughout the program, all of the learning goals will be met. You can find a convenient overview of all the different learning goals -and what activities they are acquired in- in the **matrix for primary education**.

Cfr annex learning goals

BACKGROUND

21st Century Skills

The world Economic forum made an investigation about the most needed skills in 2020 compared to 2015.

> *Creativity will become one of the top three skills workers will need. With the avalanche of new products, new technologies and new ways of working, workers are going to have to become more creative in order to benefit from these changes.*

Similarly, active listening, considered a core skill today, will disappear completely from the top 10. Emotional intelligence, which doesn't feature in the top 10 today, will become one of the top skills needed by all. (https://www.weforum.org/agenda/2016/01/the-10-skills-you-need-to-thrive-in-the-fourth-industrial-revolution/)

The World Economic Forum also mentioned that 65 % of the pupils entering school today at 6 will work in a job that doesn't exist today. This means that learning receives a total different dimension.

In Europe, the OECD defined a set of skills that children, youngsters and adults alike will need in order to thrive in the modern society: the 21st century skills. These are interconnected skills, such as problem solving, collaboration, creative thinking and computational thinking, that should be part of all education, supporting the citizens of the future and setting them up for lifelong learning.

These skills are:

1. Critical thinking
2. Creative thinking
3. Problem solving
4. Computational thinking
5. Information skills
6. Base IT-skills
7. Media literacy
8. Communication
9. Collaboration
10. Social and cultural skills
11. Self regulation

In the CODINC methodology, we explicitly aim to enhance and support these 21st century skills, focusing in particular on computational thinking, problem solving, creative thinking, collaboration and communication. The other skills are involved as well in the project, as these are all interconnected.

Computational Thinking

Why is Computational Thinking?

When you see the term "Computational Thinking", you might think it is a part of computer science. However, it is much broader than that. The term was coined by Jeanette Wing in 2006, at the Carnegie Mellon University.

You can find plenty of videos and resources on the subject, but we recommend you look at this short explanation by Google: http://bit.ly/CTbyGoogle.

Computational Thinking is not just one skill, but a range of concepts, applications, tools and thinking strategies that are used to solve problems. These can be used to discover the human DNA genome, or to analyse the writings of Shakespeare. You can practice Computational Thinking without ever even touching a computer. But what is it then?

> *"Computational Thinking is an approach to problem solving. So it is taking apart a problem and figuring out how to attack it, using what we know about computation." (Diane Main, Director of Learning, Innovation and Design, The Harker School, Upper School)*

Google defines four major facets to computational thinking:

1. Decomposition: breaking a problem down into smaller parts;
2. Pattern recognition: finding similarities and differences between the different parts, to be able to make predictions;
3. Abstraction: the ability to find the general principles behind the parts and patterns in problems;

Algorithm Design: developing the step by step instructions to solve different problems.

Why is Computational Thinking so important?

In a society that becomes increasingly complex and focused on technology, it is imperative for pupils and students to learn to think critically as well as to be able to control and create their own digital experience. Rather than consumers of digital technology, we want pupils to become the producers of it and teach them a critical understanding. Computational thinking then becomes part of a larger Media Literacy.

This is where Computational Thinking comes into play. Not only will the current digital divide increase if we ignore this need; it will shift from the usage of technology to being able to create technology. The 'haves and have-nots' will become the 'creates and create-nots'. There would be a divide between those who can control their own technology use and generate solutions to their needs; and those who simply consume without critical thought.

An advantage is that early contact with computational thinking challenges will inspire pupils to choose more STEAM-related fields of study (Science-Technology-Engineering-Arts-Mathematics), especially girls, who experience positive feedback and increased self-esteem in these fields.

Of course, there is also the vocational need. The job market will need more and more employees educated in ICT & coding. Even now there is a shortage already. But not only coding or tech jobs will need computational thinking. With Artificial Intelligence and Robotics taking over 'simple' tasks, more and more employees will be needed with creative and problem-solving skills.

Creative Thinking

The skill of creative thinking is a challenging to define: what is creativity? In the essay on 'Systematic Creativity in the Digital Realm' (Ackermann et al., 2009), being creative play is split up into three main activities: exploration, combination and transformation.

Combine - coming up with new, surprising and valuable ideas and artefacts through combining existing ideas and objects.
Explore - expanding our understanding of an area or creative domain by coming up with new, surprising and valuable ideas and artefacts.
Transform - transforming the way we see or understand the world through coming up with new, surprising and valuable ideas and artefacts.

Within the set of 21st century skills, the creative thinking skill is divided into several smaller skills, such as (1) knowing and using creative techniques; (2) leaving the beaten path; (3) seeing new connections and combinations; (4) daring to take (calculated) risks; (5) seeing mistakes as learning possibilities; and (6) having an open, investigative attitude.

Why learn to code?

There are many ways to develop 21st century skills; specifically, computational thinking and creativity. However, we believe that teaching pupils and students to code is one of the best paths to follow. This is because it stimulates their capacities to identify, extrapolate and create patterns. Pupils who learn to code have an increased understanding of systems and how they are designed. They continuously analyse problems and come up with novel solutions for them. Coding also offers pupils the tools to start creating on their own.

Understanding code and the principles behind coding languages provides a significant advantage for future employers and job-seekers. But does not mean that the advantage of coding is limited to vocational purposes.

Learning to code allows the pupils to become creators and producers in a digital world. Instead of consuming technology on a daily base, with an uncritical attitude, they can now start building their own digital future. Instead of app-consumers they become app-developers. Whether it is vocational or just for fun, learning to code provides a great advantage.

Why coding with disadvantaged students and pupils is important?

Is giving access to computers enough to reduce the digital gap between higher social classes and disadvantaged groups? Many reports highlight the need of learning to use the computer or other devices.

Access to information and the possession of the adequate information competence have become key assets in economic and social life and may give rise to a new source of stratification in the society. The "knowledge gap" is far from being a new concept (Tichenor, Donohue, and Olien 1970), but it may achieve increased importance in the digital era (Bonfadelli 2002). The pervasiveness of digital technology and ICTs has reached unprecedented levels in human life history. Information is more and more accessible on the Internet and digital devices. Per most studies, the so called first digital divide (i.e., the socioeconomic gap in access to technology) is decreasing and deemed to disappear alongside the maturation of the digitalization process. Yet, despite the popular assumption that digitalization and the Internet would lead to an overall and equal increase of knowledge in the population, recent empirical evidence points to substantial social disparities in how 4 Additional analyses including a quadratic term of the ICT possession index to capture possible nonlinearities in the relationship with navigation skills partly confirm this interpretation, as the negative association is only detected at high values of the index. Too much technology around in the home environment does not translate into better use and higher skills. Higher social classes may employ their higher resources and cultural capital to take more advantage from new technologies and hence achieve faster and better access to information.

Students of higher social classes take more advantage out of technologies than their disadvantaged peers who are mostly using the computer as a consumer playing games, using Facebook and purchasing products on the Internet.

Increasing the digital skills of disadvantaged groups is extremely urgent to reduce this digital gap and favour social integration and mobility. Producing multimedia products with mobile devices or computer, changes the relationship between the learner and the devices. When you produce some ICT content, you become a prosumer and you learn to consider the computer as a tool for life-long learning. That's also one of the indirect objectives of this project.

Using coding to elevate the position of disadvantaged groups and increase self-esteem and confidence

Giving disadvantaged students a role to play in the school environment is giving them a voice and a sense of belonging to the school community. The peer-to-peer methodology is an appropriate choice to increase their well-being and empower younger people. Young people of third-country background and refugee background may not be offered the same scientific capital or social capital as their peers, leaving them at risk of higher drop-out rates and lower self-esteem. The CODINC project offers students of disadvantaged backgrounds like refugee or third country backgrounds scientific and social capital when training them for the role of being a coding/STEAM animator to their peers. This allows role helps increase their confidence and self-believe as being actors in the digital world and leaders in their community. The CODINC experience can also improve the teacher-student relationship in the classroom as teachers have the chance to see students engaging in education in a different way and experiencing a different role. This can also help raise the level of cooperation and social cohesion in the classroom.

Core Pedagogical Principles and Facilitating

In the CODINC methodology, the core pedagogical principles are built upon the four P's forwarded by Mitch Resnick (MIT, LifeLong Kindergarten and the creator of the Scratch visual programming language): Projects, Passion, Peers and Play.

You can watch his short introductory video on Creative Learning and the four P's here: http://bit.ly/resnickfourp

Projects

The pupils and students are most motivated (and learn best) when they are working on projects they love and that have meaning to them. Therefore, the program is not limited to tutorials or following examples; but leads up to building a personal project. Pupils create their own game, one that they came up with themselves and that they get to be proud of.

Passion

Learning is fun, or at least it should be. In the 10-hour program, one of the main goals is for both the pupils and the students to enjoy themselves and love what they are doing. Being passionate about a subject or project leads to greater motivation, increased engagement and deeper learning.

Peers

It is no coincidence that the program applies a peer-to-peer approach to learning. Working, playing and working together is one of the strongest learning processes there is. Not only will the secondary students teach to their younger peers, they will also learn from them. Solitary learning is not only slower and restrictive; it is boring.

Play

This program is not composed by traditional lessons. The goal is to play with technology, with concepts and with each other. Through active experimentation and fun games, pupils will learn more than they would by sitting and listening. Therefore, the ideal atmosphere should be playful and fun; not restrictive or traditional.

The coding and the maker education movement are based on the pedagogical principles from pedagogues like Jean Piaget and Paolo Freire and they start from the following point of view:

Pupils are natural learners

Their desire to understand the world around them, to gain skill and competence in that world, and to play a meaningful part in it, is as strong as their desire for food, warmth, comfort, and love.

How do they do it? They learn through being immersed in a human culture, and through keen observation, play, imitation, and participation within that culture. They see what older, more skilled people are doing, and they have a powerful urge to do those same things. Indeed, little pupils explode with rage when they are not permitted to do things they see others do. They want to join in that dance called "life," not just sit on the sidelines. Pupils are like scientists; they develop theories, make hypotheses, test them, and revise or abandon their theories as necessary.

Teaching can help learning when it genuinely supports and enables people to do what they want to do, when it helps them figure out whatever they're trying to figure out - but only when such an intervention is wanted, asked for, invited, or in some way accepted by the learner. Teaching that is uninvited, unwanted, and unasked for does not help learning. It hinders it.[1]

Facilitating coding and offline activities

Making and creating can be challenging. To support creativity and innovation, we have to help students view these challenges as a normal part of the process and one they can learn to navigate successfully with practice. To teach coding, it helps to understand key stages of the process, students' reactions at each stage, helpful teacher responses, and skills taught.

This approach can transform the traditional classroom mindset where teachers passes knowledge in a less traditional way of working where the teacher becomes an animator coaching their pupils to learn.

A single summative evaluation of the learned knowledge can change into a series of smaller assessments using student-created products that build on one another and culminate in a more substantial final project.

During the unplugged and coding activities, the pupils and the students are stimulated to try some things and to see if they work. Trial and error can lead to failure but also iteration and innovation.

All students can be challenged and can grow, and that in pursuit of this goal, we build a safe space where failure is a fundamental possibility. A failed experiment is not an end, as with a standardized assessment, but rather a challenge to try again.

When students are free to invent and create, they begin to see technology as a means for solving real-world problems and taking their learning to the next level.

Our resources are designed specifically as hands-on learning tools to help today's students build skills for the creative and digital economy — critical thinking, collaboration, communication, curiosity, problem solving and invention.

[1] What We Mean By LEARNING, lego dacta
http://learn.media.mit.edu/lcl/resources/readings/what-we-mean-by-learning.pdf, consulted on 26/03/2018

Do I, as a teacher, must know how to code before the start of the activities?

Firstly, during this project, the trainers will train the students and their teachers in the basic of coding. The trainers will come for a 10 hour of code in the classroom. Afterwards, the students of the secondary school are going the primary schools.

The teachers of the primary schools are trained to facilitate the students together with the pupils. So, you don't need to know how to code before start of the project. You should be willing however to try and experiment with a new learning approach in the classroom. If a student finishes early, invite them to either improve on the project they made (by adding a score, increasing the difficulty, changing the theme of the game …) or to create a second, more difficult one.

At the end, allow for 10 minutes so that students can test each other's projects and have a quick discussion about what they learnt so far.

THERESHOLDS

During the Capital Digital project, the best pratice on with the CODINC methodology is based on, schools were supportive to allow for free coding activities if the materials and hardware to teach coding were provided. It is important that trainers carrying out the CODINC project have a good and structured preparation with the teachers and schools to best align learning outcomes and schedules for the students prior to starting the project.

Teachers

Teachers have been supportive to use the Capital Digital project once they see what the activities are and once they see that the pupils are enthusiastic and enjoying the activities.

It was also found that school directors and principles were proud to provide coding activities in the classroom: coding is in demand and parents like the fact that coding is introduced in the school. The introduction of coding and computational thinking methodologies helped raise the prestige of the school. This is excellent for schools that have been identified to be in excluded areas. It is important that the trainers speak with the director prior to starting the CODINC project in the school to get their support in a written way by making an agreement.

Say no to chaos in the classroom, bring some structure

Teachers are afraid of chaos in the classroom, because in their eyes non-organization is an obstacle for the teaching of pupils. That's why the exercises need to be given in a very structured way. It is important to think about the way you organize the classroom and how the different exercises will be scheduled. In our training curriculum with the students we should discuss this. Can the structure of the classroom be changed for this day or not?

How can me make group work with pupils without changing the structure of the classroom? Is there another room that can be used for the activities?

This is an important thing to discuss with the teachers before the start of the project.

Discuss the timetables before you start with the project

For the coding activities it is interesting to have some blocks of two-three working hours, instead of courses from 50 minutes like custom in most secondary schools.

5 blocks of 2 hours or 2 blocks of 3 and 2 of 2 are ideal to plan the activities with the students. In primary school, it is easier to schedule this than in secondary.

Pupils

Show a try out in the classroom, before you start the project

Introduce the coding activity in the classroom by yourself, telling the pupils what they can expect from the activities in this project and how the project will be developed.

Let them try out something practical, so they see that this is fun. Most of the pupils are glad when there are projects in the classroom, it changes from the daily routine and dry courses they get.

Ask them about their fears for working with the pupils and discuss this with them.

If you have the possibility, try to look with them what kind of primary school they want to work with. If not explain them why the school you choose, need their help for learning the pupils to code.

Role models for little brothers and sisters

In most of the disadvantaged families, ties between siblings are very strong and "big" brothers and sisters like to help the little ones. Talk about being a role model for the little ones and the role they can play in the fact that pupils will discover coding.

Annex 1 Matrix of learning outcomes

WORKSHEETS

https://docs.google.com/document/d/1wEyC7jfkImP7ks15qr_cx7pFo7kqc5Y0X0lhqYg5U6c/edit#

LINKS

Lightbot (online en apps) - http://lightbot.com/hour-of-code.html

Hour of Code: Star Wars - https://code.org/starwars

Hour of Code: Minecraft - https://code.org/minecraft

MATERIALS

Item	Estimated price / item	Number	Estimated price total €
Makey Makey set	Around 55 euro for a set	10	550
Computers / laptops		10	
Tablets (optional)	100	10	1000
Crafting materials			
Writing materials			

APPENDIX I: REFERENCES

Ananiadou, K. (2009). *21st Century Skills and Competences for New Millenium Learners in OECD Countries*.

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design*. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada.

Raspberry Pi Foundation. (2017). *CodeClub Curriculum.* Retrieved March 6, 2018, from https://codeclubprojects.org/en-GB/curriculum/

http://curriculumvandetoekomst.slo.nl/21e-eeuwse-vaardigheden

The second digital divide in Europe. A crossnational study on students' digital reading and navigation skills* Davide Azzolinia† , Antonio Schizzerottoa,b

https://irvapp.fbk.eu/wp-content/uploads/2017/09/FBK-IRVAPP-Working-Paper-No.-2017-02.pdf