CLEI electronic journal, Volume 21, Number 2, Paper 9, August 2018

The “Code Yourself!” and “¡A Programar!” programming MOOC for teenagers: Reflecting on one and a half years of experience

Inés Friss de Kereki, J. Víctor Paulós
Universidad ORT Uruguay, Facultad de Ingeniería,
Montevideo, Uruguay, 11100
kereki_i@ort.edu.uy, paulos@ort.edu.uy

and

Areti Manataki
University of Edinburgh, School of Informatics
Edinburgh, United Kingdom, EH8 9AB
A.Manataki@ed.ac.uk

Abstract

Critical thinking and problem solving are fundamental skills to function successfully in today’s world. When programming, these skills are promoted, developed and deployed. In this context, Universidad ORT Uruguay and The University of Edinburgh co-created in 2015 a MOOC (Massive Open Online Course) that teaches young teenagers how to program. The course was offered simultaneously in two versions: in Spanish, called "¡A Programar!" and in English, called "Code Yourself!", which are available on the Coursera platform. Since its launch in March 2015, more than 161,000 people from 197 countries have registered. Initially it was offered in a "fixed session"; while currently it is offered in an "auto-cohort" mode. In both cases, student surveys indicate that the course has met or exceeded expectations (values above 93%). In this paper, we detail the characteristics of the MOOC, and we analyze and compare the results for the two delivery modes.

Keywords: MOOC, programming, computational thinking, Scratch.

1 Introduction

The world has changed so fundamentally in recent decades that teaching and educational roles are no longer the same [1]. It has been argued that in order to function in today’s society, every citizen must understand at least the principles of computer science [2]. Coding skills can help understand today's society and foster 21st century skills like problem solving, creativity and logical thinking [3]. There is a growing interest to equip young people worldwide with coding skills. MOOCs (“Massive Open Online Courses") are a recent development in the educational landscape, widening access to education and democratizing knowledge, through the free delivery of high-quality academic content on a grand scale. Even though there is a wide range of MOOCs in computer science, there are only a few programming MOOCs designed for children and teenagers. Recognizing this gap, Universidad ORT Uruguay and The University of Edinburgh have jointly designed a bilingual MOOC that teaches young people how to program: the course is called “¡A Programar!” [4] in Spanish and “Code Yourself!” [5] in English, and it is available on the Coursera platform. It was first offered in March 2015 in a “fixed session” mode, while since August 2015 it has been available on a “flexible session” or “auto-cohort” mode.

In this paper, we describe the design of the course, we present the characteristics of the two modes in which it has been offered, and we compare results. We begin by discussing concepts related to computational thinking and the inclusion of programming and computer science in the current curricula (Section 2). Existing resources for learning how to program are next presented in Section 3. In Section 4, we present the state of the art in MOOCs, including platforms, design recommendations and courses specific to computer science. In the next section (Section 5), we describe the joint design of “¡A Programar!” and “Code Yourself!”, and we discuss the considerations made when developing course materials for a younger audience. Section 6 presents and analyzes the two modes in which this MOOC has been offered. Results are presented in Section 7 and, finally, conclusions and future work are discussed in Section 8.

2 Computational Thinking

Critical thinking and problem solving skills are essential skills that students must acquire in order to succeed in today's world [6]. Over the past decade, Information and Communication Technology (ICT) education has transitioned from focusing on ICT as a tool toward understanding the underpinning concepts and working of digital technologies [7]. Currently, jobs requiring solving unstructured problems, communication, and non-routine manual work have grown as a proportion of the labor market, so young people's preparation for the workforce should adapt accordingly [8]. Computational thinking (CT) is a fundamental skill for everyone. It involves solving problems, designing systems and understanding human behavior, for instance through the use of abstraction and decomposition [9]. In particular, CT is a problem solving process that includes representing data through abstractions such as models and simulations, identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources, and generalizing and transferring this problem solving process to a wide variety of problems [10]. CT infiltrates all areas, both sciences and humanities, and it is already changing the way we think [11]. Hence, in an increasingly information-based society, CT is becoming an essential skill for everyone [12].

At the same time, the importance of acquiring coding skills is becoming widely recognized. It has been argued that "coding is not a set of technical skills but a new type of literacy and personal expression, valuable for everyone" [13]. It is a new way for people to organize, express, and share their ideas [13]. Programming is not only a fundamental skill of Computer Science and a key tool for supporting the cognitive tasks involved in CT, but also a demonstration of computational competencies [14]. In fact, coding skills are the most visible part of computational thinking [15]. Teaching computer science should include coding combined with computational thinking, thus introducing ways to process and represent information, work systematically to identify and eliminate errors, as well as divide and solve problems [16].

Computing education has evolved over the past decades. Historically, computing education started in universities [17]. Nowadays, programming and computer science has been introduced into early childhood education in several countries and there are plans for introducing it in even more [18]. Some examples are presented by the European Schoolnet [3], according to which there are 16 countries that include coding at the national, regional or local curriculum (England, Spain and France among others) [3]. In England, in particular, the national curriculum for computing aims to ensure that all pupils can understand and apply the fundamental principles and concepts of computer science, as well as analyze problems in computational terms [19]. Similarly, computer science learning standards in the United States indicate that fundamental concepts of computer science should be introduced to all students, beginning at the elementary school level [2].

3 Initiatives and resources for learning to program

There is a growing number of initiatives and resources available for young people that wish to learn how to program. Code.org [20], for example, provides several videos and short lessons for building computer programs by snapping programming blocks together. CodeAcademy [21] is an interactive platform for teaching professional programming languages, such as Python or Java. There are also programming languages particularly designed for young people, such as Scratch [22], Alice [23], and AppInventor [24]. These languages allow for "drag-and-drop programming", and they are particularly beneficial for younger students because they require less typing and are more engaging and visual compared to industry-standard, textual-based languages [25].

In particular, Scratch is a visual programming language developed by the MIT Media Lab. It allows to create animations, games and stories. It is aimed at children between 8 and 16 years, and it is designed to be fun, educational and easy to learn [22]. Through the use of Scratch, computing concepts such as sequence, events and iterations, and computing practices such as abstraction and modularization are applied. All these elements are related to computational thinking [26]. For example, the use of abstraction involves deciding scenes and characters to include or not in the stories and games created in Scratch. It is an example of application of computational thinking [8]. In addition to the programming environment, Scratch includes an active online community, with over 14 million registered users and 17 million shared projects [22]. Programming in Scratch and sharing interactive projects provide the opportunity to learn important mathematical and computational concepts, as well as how to think creatively, reason systematically, and work collaboratively, all essential skills for the 21st century [27]. However, in order to make the most of this opportunity, it is important to provide the appropriate instructional scaffolding. MOOCs may be particularly useful in this context.

4 MOOCs

A MOOC is an online course, massive and open, offered over the Internet and available for free to a very large number of people [28][29]. “MOOCs are a recent expansion in e-learning and distant education that have experienced rapid development and achieved substantial attention from a broad range of learners” [30]. Some reasons are: sufficient bandwidth for streaming video, improved web technologies, and platform-related advances [29]. MOOCs have the potential to provide education on a global scale [31] and it is this large scale that differentiates MOOCs from traditional distance education [32]. MOOCs also provide students with free access to high quality academic content [33]. They assist in developing skills for collaborating with others online and developing digital artifacts, which are essential digital economy skills [34].

There are several platforms that offer MOOCs. Coursera [35], edX [36] and Udacity [37] are among the most recognized ones, while in Spanish it is worth mentioning Miríada [38]. Coursera offers more than 1,600 courses [35], edX more than 900 [36], Udacity more than 140 courses [37], and Miríada more than 375 courses [38]. Regarding number of students, Coursera has more than 15 million students, edX more than 5 million students, Udacity indicates 4 million users [39], and Miríada about 2 million students [38]. These large numbers are indicative of the global outreach that universities can have through MOOCs.

It is also worth mentioning that MOOCs can be effective means to educate students when face-to-face, on-campus teaching is not possible or available [31]. They can also complement and enrich traditional face-to-face teaching, allowing a variety of styles of classes [40]. For example, Siever [29] discusses three ways in which traditional educators can leverage the strengths of MOOCs to benefit their students: MOOC resources and content can be used for “flipped” classrooms; they can provide a way to study alternative approaches to teaching; and they can support independent study.

There has been a big debate over the last few years about the value that MOOCs bring to the educational landscape. The initial hype around MOOCs was followed by skepticism, with the debate still being in progress [41]. One of the main criticisms involves the initial claim that MOOCs can improve access to higher education. Current evidence shows that the average MOOC learner is university-educated, middle-aged, middle-class and from the developed world [42], and hence several scholars have indicated that “MOOCs are falling far short of democratizing education” [43]. At the same time, others have argued that “among the millions of learners who have taken MOOCs, there are some for whom this is their only way to access a rigorous, college-level course” [44]. Concerns have also been raised regarding the sustainability of MOOCs [41][42][45]. As Sharrock points out, “while MOOC development costs are quite high, the lack of a stand-alone business model poses the risk that non-credit MOOCs, whatever their quality, will collapse without continuing support from investors and institutional sponsors” [41]. MOOC providers have, therefore, been exploring different options for sustainability. Coursera, for instance, has recently introduced fees for earning grades in many of its courses, while monthly subscriptions are also offered to learners taking a specialization. Questions have also been raised about the pedagogical approach in MOOCs, and it has been argued there is a focus on direct knowledge transmission and learning materials rather than dialogue and critical thinking [42] [46]. Finally, much discussion has taken place around high drop-out rates in MOOCs, with some scholars interpreting them as a sign of failure and others indicating that many learners enroll in MOOCs simply to browse content rather than complete the course [41].

With MOOCs being a recent development, the community is still experimenting and exploring how to design successful courses. Some recommendations by Button [40] include keeping courses short (up to four weeks), keeping videos short (no longer than six minutes) and having course instructors project an informal air, seated at a desk instead of standing at a podium. Alario et al [47] suggest the creation of video lectures between 5-10 minutes and the combine of different resources, such as digital drawings, “talking head” with slides and, animated slide presentations [47]. According to Guo et al [48], videos where instructors speak fairly fast and with high enthusiasm are more engaging. Hence, their recommendation is to coach instructors to bring out their enthusiasm and reassure them that they do not need to purposely slow down [48]. As far as success factors are concerned, the more satisfied a student is with the course instructor, the teaching material, and the assignments, the more probable it is that he/ she will successfully complete the course [49]. Moreover, peer assessment has a positive effect on the probability to successfully complete the course [49].

There is a variety of MOOCs that teach the basics of computer science. Most of them use programming languages for professional use, such as Java or Python, and they are typically addressed to adult learners. For example, the 12-week MOOC "CS50x: Introduction to Computer Science" [50], which is available on the edX platform, teaches how to think algorithmically and solve problems efficiently. It uses multiple programming languages (e.g. C, PHP, JavaScript and Scratch), and topics covered include: abstraction, algorithms, data structures, encapsulation and security. In our opinion, this is a rich course, both breadth- and depth-wise, but it can be challenging for young people and complete beginners to follow it, particularly because the level of difficulty grows rapidly throughout the 12 weeks.

"Intro to Computer Science" [51], is a 3-month MOOC available on Udacity, which uses Python. The focus is on building a search engine and a social network. According to Ben-Ari [52], the course provides “a good level of scaffolding, so the student is not challenged to build a program from scratch”. Moreover, it does not cover software engineering practices explicitly, which is an important element of computer science education. Another example of a MOOC that teaches introductory computer science is "Computer Science 101" [53]. This course, which is available on Coursera, teaches the basics of computing and it covers fundamental ideas, such as abstraction, logic and loops. It uses a variant of the JavaScript programming language, but the focus is not on programming itself, but rather on general concepts in computer science.

Recently there have been some efforts towards developing computer programming MOOCs particularly for young people. An example here is "MyCS: Computer Science for Beginners" [54], which is offered by Harvey Mudd College. This course explores how computers work and how we can use them to solve interesting problems. It lasts 5 weeks, throughout which lessons alternate between general topics in computer science and programming activities. It consists mostly of text guides, with only a few videos included, which are not of high-quality production, as recognized by the authors of the course. We believe that these factors could be demotivating for young people. “Programming in Scratch" is a 6-week MOOC offered by the same institution and addressed to a young audience [55]. The focus of this course is on programming in Scratch, and hence it does not cover topics in computational thinking or software engineering.

In Spanish there is less availability. It is worth mentioning "Computational Thinking in School", a 5-week MOOC available on Miríada [56], which consists of two parts: a) conceptual introduction to computational thinking and its application in daily life, and b) practical introduction to Scratch. Although the course promotes a deep reflection on the concepts of computational thinking, the video content can be unattractive to young people, since it consists mostly of image presentations with voiceover. Furthermore, Scratch examples are often presented first in full and then split into their components, from which the program is rebuilt. In our opinion, this approach does not promote the problem-solving skills that we want young people to develop. "SM4T: Scratch MOOC for Teens" is another Spanish-speaking MOOC offered to young people through Plan Ceibal Uruguay [57]. It teaches fundamental programming concepts through coding examples in Scratch, but it does not cover software engineering practices. It is also specific to Uruguayan students, so it does not allow for worldwide access.

In summary, there is a plethora of MOOCs addressed to adults that wish to learn the basics of computer science and programming, but only a small number of computing MOOCs designed for younger audiences. MOOCs oriented to young people should include interesting and fun topics, different types of evaluation (e.g. peer review), as well as attractive and well-designed videos. The task complexity in these MOOCs should be appropriate for young learners and in accordance with the course level aimed. To the best of our knowledge, none of the existing MOOCs have all these features. Furthermore, none of them teach programming in conjunction with computational thinking and software engineering.

5 Designing the ¡A Programar!” and “Code Yourself!” MOOC

Universidad ORT Uruguay and The University of Edinburgh have co-developed a MOOC that introduces young people to programming, while addressing the gap described above. The course is 5 weeks long and it is available in both Spanish and English, called “¡A Programar!” [4] and “Code Yourself!” [5], respectively. Our goal with this MOOC is to introduce youngsters to programming, while promoting the development of computational thinking and the use of basic software engineering practices. The programming language used in the course is Scratch, as it was deemed the most appropriate one for young learners.

The course was designed jointly by the two universities, while taking into account the MOOC design recommendations presented in Section 4. The two teams agreed in advance which topics, examples and applications were to be used in each unit. Decisions around course assessment and communications were also jointly made by the two teams.

The course syllabus is detailed in the following five units, one for each week of the course:

- “Your First Computer Program”: This unit introduces the notion of algorithms and control structures, such as sequence, selection and iteration. It promotes developing programs in an incremental fashion and it highlights the importance of testing. Some Scratch programming components (e.g. motion and looks) are also introduced.

- “Code Gone Loopy!”: Here we continue working with control structures, focusing on different types of iteration. New concepts introduced include decomposition, event-driven programming, interface design, and software requirements. Scratch elements used in this unit are from the Control palette, Events, Pen, Sound and Sensing, among others.

- “Remixing Games”: This unit is centered on creating new versions of existing games. The concept of nested loops is introduced, and software engineering practices regarding testing and documentation are discussed. The Data and Operators palettes in Scratch are used.
- “Reusing Your Code”: Procedures, parameters and generalization are presented in this unit as part of code reuse practices. Software modularity and flexibility are also discussed. As far as Scratch elements are concerned, we introduce the use of cloning, the “More Blocks” palette and the “backpack” in Scratch 2.0.
- “Think Like a Software Engineer”: Through the step-by-step development of a complete game, we discuss in this unit different software engineering stages: requirements gathering, design, implementation and testing. The concepts of computational complexity and concurrency are also introduced. In Scratch, we present how messages can be broadcasted and how projects can be shared on the Scratch website [22]. Finally, we make suggestions for learning new programming languages.

When developing the course materials, we made careful considerations regarding the target audience. The Scratch code examples, for instance, were chosen so that they are appealing to young people, both men and women. Some examples of programs developed in the course are: representing a dance, making a short movie, creating a karaoke program, and building several fun games, such as “Fruit Ninja” (a game where the player swipes their hand across the screen attempting to slice a fruit), breaking bubbles with your hands and “chasing zombies”. Another fun element that we decided to include in the videos is “Cody”, an animated character that looks like an alien (see Fig. 1). Cody interacts with the teachers to raise questions or comments, and it is meant to keep the interest of the young student. More importantly, Cody provides opportunities for reflection, as the questions raised by the character are always designed to extend the material presented or make some clarification.

Figure 1: Image of “Code Yourself!” teacher and “Cody”

Each unit includes multiple downloadable short videos (between 3 and 5 minutes long, in general) with optional closed captions. Simple quizzes are embedded in the videos, in order to encourage student engagement [29]. The Scratch code examples presented in the videos are also provided for download, along with useful resources, such as images, sounds or links. Video interviews with experts in computing and other areas also included. The aim is to establish links between the concepts presented in the unit and practical life. For example, Unit 1 includes an interview with an audiovisual producer in a television studio, who explains the filming process, thus highlighting links to the computational terms of sequence and selection. Another example is in Unit 3, where a Biotechnology expert describes her daily tasks. From this interview, it can be identified -in another context- the concept of algorithm and the refining process or "remixing" of a process, i.e. adjusting and improving according to results. Half of the interviews were conducted in Spanish and half in English, and the subtitled videos were shared between the two course versions, thus strengthening the link between “¡A Programar!” and “Code Yourself!”.

An important course element is the discussion forums, as they help clarify student questions and build the sense of a learning community. Our common approach in the two MOOC versions is to promote student participation in the forums, while keeping a discreet teacher presence (i.e. intervene mostly in urgent cases or for clarifications regarding course logistics). Student participation in the forums is not compulsory but it is encouraged by the course instructors, for instance by mentioning in the videos some interesting topics to discuss. For example, students are prompted to develop an algorithm for teaching Cody how to brush his teeth, to present examples of abstraction in famous paintings, and to identify real-world examples of iteration. Typically, students make their contributions in a new thread, and then other students reply with their own proposal or with corrections or suggestions on the original idea.

Assessment was designed with the aim of encouraging analysis and reflection, while providing the opportunity to gain practical programming experience. It consists of five multiple choice quizzes (maximum 50 points) and two peer-evaluated programming projects (maximum 50 points). In order to successfully pass the course, students need to obtain at least 50% of the points. The quizzes can be attempted multiple times, and the highest score obtained is the one awarded. Feedback is provided not only as "right answer" or "wrong", but clues or suggestions are also provided in the case of a wrong answer. These features (multiple attempts and giving prompt feedback) are known to encourage active learning [58]. The first programming project involves creating an animation, in which two characters interact and present characteristics of a country or city of interest, such as monuments, sports or typical food. The second project is a game, in which the player helps Cody to go back to his planet, by navigating his spaceship with the use of the arrow keys, while avoiding meteorites. This project includes several advanced elements, such as counting the number of meteorite crashes and keeping track of time. Game instructions and different sounds and backgrounds should also be included, so as to achieve the animation effect. Clear instructions are provided for both projects, for instance regarding characters, backgrounds, dialogues, and sound requirements. Students are expected to upload their program and then, the platform assigns them three projects of their classmates that they need to evaluate. A detailed rubric with quantitative guides and qualitative comments is given for this task.

We should highlight that the course was designed through close collaboration between the two universities. The final version in Spanish (videos, questionnaires, etc.) was developed by Universidad ORT Uruguay and the final version in English by The University of Edinburgh. Nevertheless, the same filming guidelines were followed, so as to keep the two versions consistent. For instance, we decided to use interesting but non-distracting backgrounds, and to wear long-sleeved clothing for cultural reasons. Both Spanish and English versions used several filming formats: for example, the teacher being on the right of the screen to include text or images on the left (see Fig. 2), the teacher being in the center when developing programs, and the use of "transitions" (short cuts image) in the videos to highlight closing points. The recommendations for the design of videos presented in Section 4 were taken into account, as were the suggestions of audiovisual communication specialists. An opening and a closing video with both teachers was included in both "¡A Programar!" and "Code Yourself!", so as to further strengthen the idea of a single course with two versions.

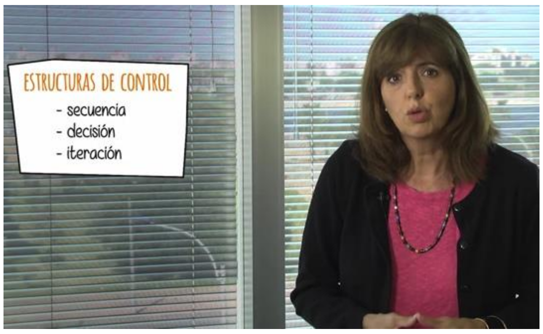


Figure 2: Image of "¡A Programar!"

All additional materials for the videos were also shared in both courses. This includes resources such as "stop-motion" (an animation technique that physically manipulates an object so that it appears as if it is moving on its own), graphics, pictures, sounds and cards with definitions and algorithms.

This international collaboration was enabled through video-conferencing and collaborative tools, such as Google Docs. A 3-day face-to-face meeting also took place in Edinburgh, which strengthened the links between the two teams and helped establish collaboration strategies. In addition to the course instructors, the production team included graphic designers, education experts, audiovisual producers, editors, communication specialists, makeup artists, and educational platforms experts and coordinators.

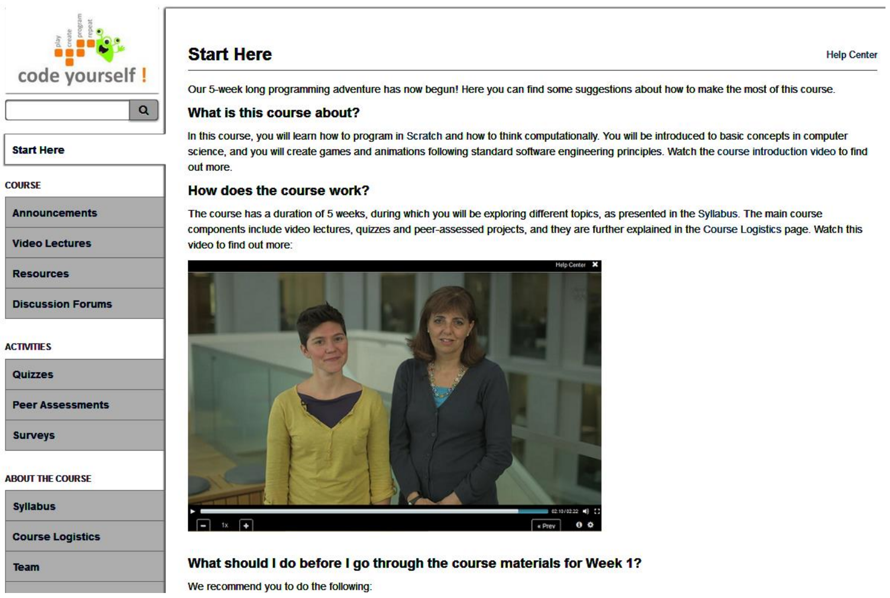
6 Sessions and modes

"¡A Programar!" and "Code Yourself!" have been available on the Coursera platform since March 2015. The course was announced and promoted through various channels: social media, educational institutions and through the Coursera website. Prior to its first launch, the course was intensively promoted to young people via Facebook and Google Ads. The first course session was between March and April 2015. From August 2015 onwards, the course has been available in an "auto-cohort" or "flexible" mode, in accordance with related changes in the Coursera platform. This flexible mode means that new sessions beginning each month and it is possible for students to easily switch between different sessions while transferring with them any coursework already done.

In this section, we describe the features of the two delivery modes on the Coursera platform in relation to the different course sessions. As already mentioned, these features include aspects related to session start and end dates, the Coursera website interface, coursework submission and evaluation dates, and certification. There were also differences regarding teacher participation and the use of the discussion forums.

6.1 First session: March/April 2015

The first session of the course took place between March and April 2015, in "fixed" session format, that is, a single session with preset and fixed dates. In this first session, the Coursera platform had certain characteristics specific to this format, which we will detail. The interface looked very similar in both languages, as shown in Fig. 3 and Fig. 4. A screenshot of the course website in English is provided in Fig. 3, which depicts the course structure, consisting of "Announcements", "Video Lessons", "Resources" and "Discussion Forums". Under "Activities", one can find the course quizzes, surveys and peer reviewed projects. General course information is provided in "About the course", covering topics like the course syllabus, format and assessment, as well as faculty and production team.



Start Here

Our 5-week long programming adventure has now begun! Here you can find some suggestions about how to make the most of this course.

What is this course about?

In this course, you will learn how to program in Scratch and how to think computationally. You will be introduced to basic concepts in computer science, and you will create games and animations following standard software engineering principles. Watch the course introduction video to find out more.

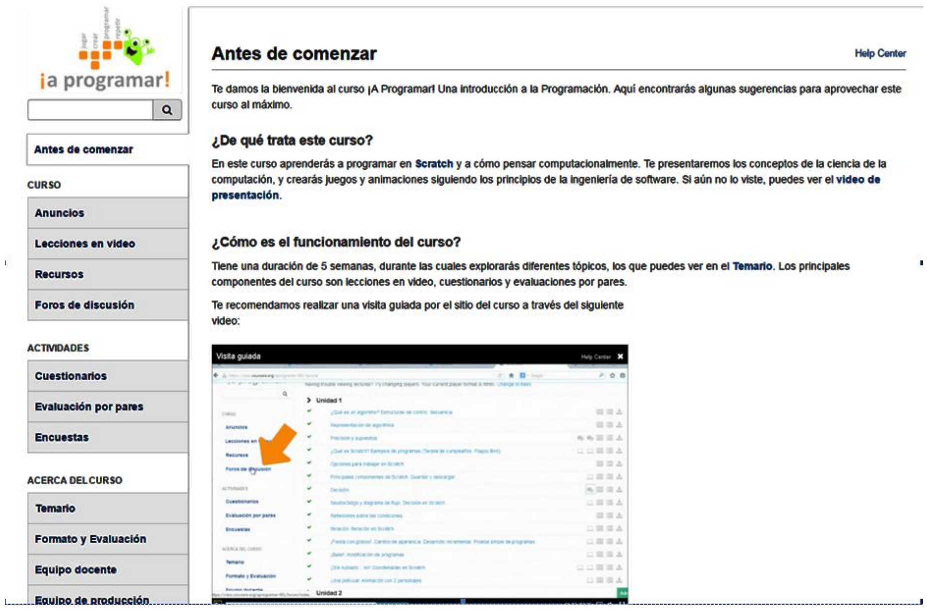
How does the course work?

The course has a duration of 5 weeks, during which you will be exploring different topics, as presented in the Syllabus. The main course components include video lectures, quizzes and peer-assessed projects, and they are further explained in the Course Logistics page. Watch this video to find out more:

What should I do before I go through the course materials for Week 1?

We recommend you to do the following:

Figure 3: Website of the first session of “Code Yourself!”



Antes de comenzar

Te damos la bienvenida al curso ¡A Programar! Una introducción a la Programación. Aquí encontrarás algunas sugerencias para aprovechar este curso al máximo.

¿De qué trata este curso?

En este curso aprenderás a programar en **Scratch** y a cómo pensar computacionalmente. Te presentaremos los conceptos de la ciencia de la computación, y crearás juegos y animaciones siguiendo los principios de la ingeniería de software. Si aún no lo viste, puedes ver el **video de presentación**.

¿Cómo es el funcionamiento del curso?

Tiene una duración de 5 semanas, durante las cuales explorarás diferentes tópicos, los que puedes ver en el **Temario**. Los principales componentes del curso son lecciones en video, cuestionarios y evaluaciones por pares.

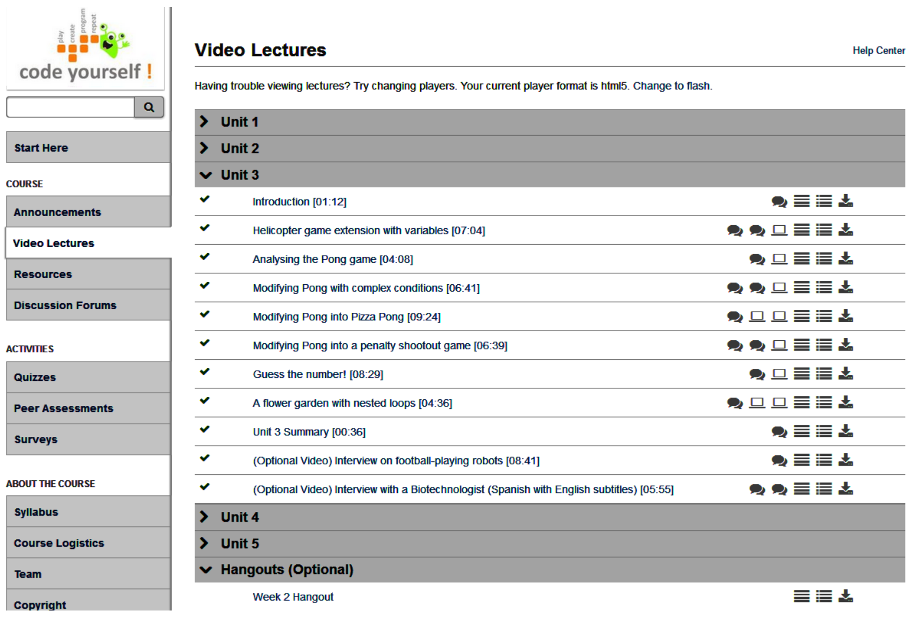
Te recomendamos realizar una visita guiada por el sitio del curso a través del siguiente video:

Figure 4: Website of the first session of “¡A Programar!”

Fig. 5 presents the detailed list of videos for each unit in “Code Yourself!” (in “¡A Programar!” it is similar). Each video includes its title, duration, and resources available, such as specific discussion forum/s, associated code examples, subtitles and the option to download.

Each week a new unit was offered (5 weeks in total). The submission deadlines for quizzes and projects were preset and fixed, while late submission was not allowed. It is also worth noting that, even though assessment was based on 5 quizzes and 2 peer-reviewed projects, attempting each of these tasks was not compulsory. All that was needed for students to pass the course, was to get 50% of the total points. This means, for instance, that a student could pass the course only by correctly answering all the quizzes. Peer assessment in the fixed session format was anonymous: students did not know the identity of the students they were evaluating or being evaluated by.

Upon successful course completion, students received a “statement of accomplishment” (i.e. a certificate) from Coursera. This was provided for free in a pdf format, and it was signed by both course instructors. For students obtaining 90% or more of the total points, the certificate also indicated that a "distinction" was awarded. The free certificate did not verify the student identity. A verified certificate was available as a paid option.


code yourself !

Start Here

COURSE

Announcements

Video Lectures

Resources

Discussion Forums

ACTIVITIES

Quizzes

Peer Assessments

Surveys

ABOUT THE COURSE

Syllabus

Course Logistics

Team

Copyright

Video Lectures Help Center

Having trouble viewing lectures? Try changing players. Your current player format is html5. Change to flash.

- Unit 1
- Unit 2
- Unit 3
  - ✓ Introduction [01:12]
  - ✓ Helicopter game extension with variables [07:04]
  - ✓ Analysing the Pong game [04:08]
  - ✓ Modifying Pong with complex conditions [06:41]
  - ✓ Modifying Pong into Pizza Pong [09:24]
  - ✓ Modifying Pong into a penalty shootout game [06:39]
  - ✓ Guess the number! [08:29]
  - ✓ A flower garden with nested loops [04:36]
  - ✓ Unit 3 Summary [00:36]
  - ✓ (Optional Video) Interview on football-playing robots [08:41]
  - ✓ (Optional Video) Interview with a Biotechnologist (Spanish with English subtitles) [05:55]
- Unit 4
- Unit 5
- Hangouts (Optional)
  - Week 2 Hangout

Figure 5: Detailed list of videos and other materials for each unit in “Code Yourself!”

Teacher engagement during this first session was high, both by the course instructors and teaching assistants, who showed presence mostly through weekly announcements on the course website and via email. At the beginning of each week, an announcement publicized with the news of that week, detailing the topics to be covered. Other logistics details were also included, such as project submission deadlines. For instance, a project submission deadline extension was once granted and announced due to an unexpected Scratch website unavailability. As far as discussion forums are concerned, teacher presence remained low, as agreed at the course design stage.

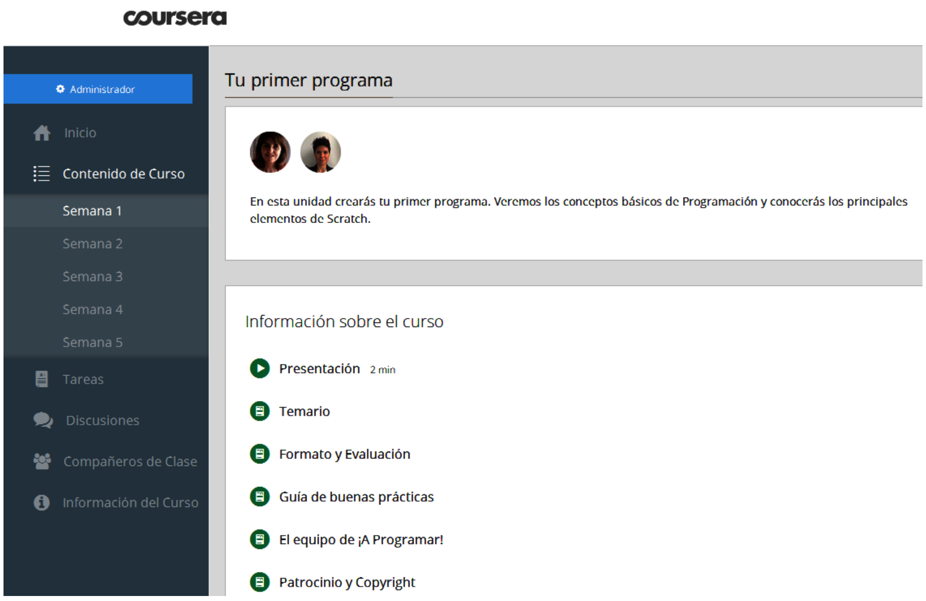
6.2 Following sessions: since August 2015

The first course session helped us identify areas for improvement, based on student feedback received, discussions in the forums and joint analysis of the teaching and learning experience by members of the teams in Uruguay and Edinburgh. It was decided to maintain the focus and content of the course. Some minor adjustments in assessments were decided, namely to modify the minimum scores required for each assessment task, which were established at 50% for each quiz and peer-assessed project. Some quiz questions and answers were also improved, based on student feedback. Furthermore, we decided to include some new resources, such as documents with information on how to upload and download files, as related questions were very frequently asked during the first session of the course.

As already mentioned, “¡A Programar!” and “Code Yourself!” sessions were offered in an auto-cohort mode from August 2015 onwards, with several changes introduced by Coursera. The most important changes involved student enrollment, communications, course website interface, assessment and certification, and they are described hereafter. As far as enrollment is concerned, in the new delivery mode, students can enroll for the current session or next session, with sessions beginning each month. They are allowed to enroll to an ongoing session within 11 days since its start date. If enrolled earlier than the start date, the content for the first week of the course (including Quiz 1) is available, so as to motivate students to engage as soon as possible. As soon as the new session begins, the content for all units becomes available.

Regarding communications, Coursera currently sends an email at the beginning of each week to learners enrolled in a course, reminding them about upcoming deadlines and content. We have therefore decided to drop our weekly announcements (as sent in the first session of the course), so as not to overload students. Instead, we included an introductory text to the webpage for each week.

The interface of the Coursera platform has been redesigned and simplified for auto-cohort courses, as shown in Fig. 6. The content is organized by weeks (similarly to "units" in the previous interface). It is worth mentioning that the new Coursera interface for both learners and course instructors is continuously adjusted and improved. For instance, during the first few sessions in this new modality, the Coursera platform showed the number of classmates in the running session and their geographical distribution (without providing student personal details), in order to generate a sense of community and belonging (Fig. 7). In last few months, however, this information is no longer available.



Tu primer programa

En esta unidad crearás tu primer programa. Veremos los conceptos básicos de Programación y conocerás los principales elementos de Scratch.

Información sobre el curso

Presentación 2 min

Temario

Formato y Evaluación

Guía de buenas prácticas

El equipo de ¡A Programar!

Patrocinio y Copyright

Figure 6: Website of "¡A Programar!" in the flexible mode

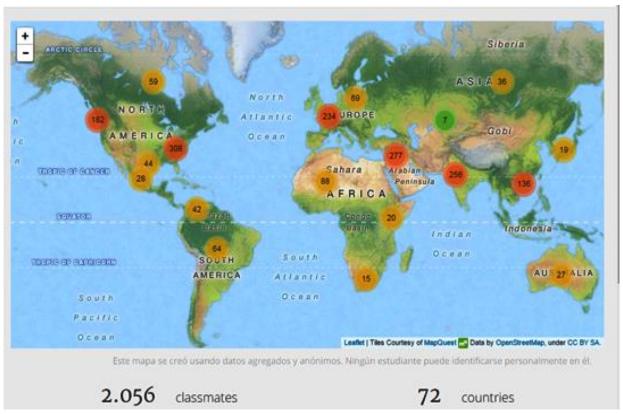
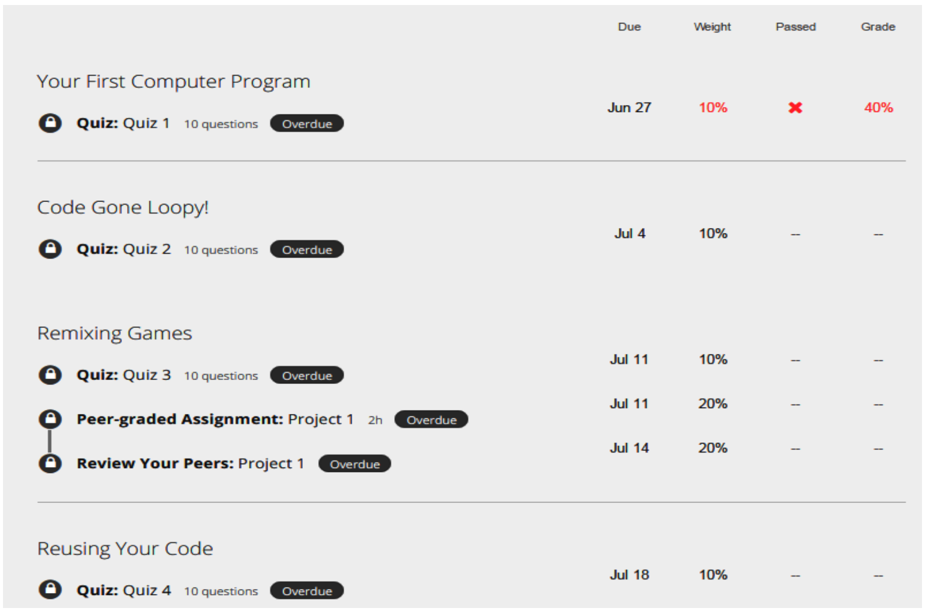


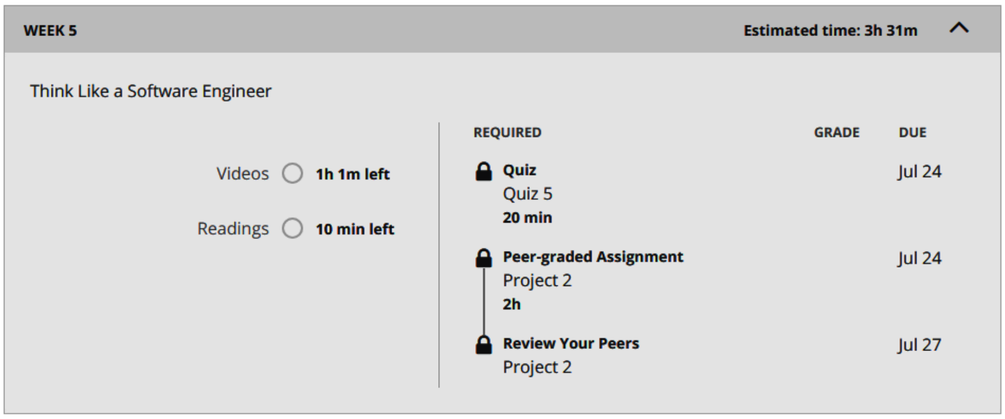
Figure 7: Classmates and countries (screenshot from the "Code Yourself!" website)

Furthermore, information is provided about the learner's personal progress and estimated time to complete each week's content. For example, the screenshot in Fig. 8 depicts a student's personal progress regarding assessments. It makes clear that in Week 1 (i.e. the week entitled "Your First Computer Program") there is a quiz with 10 questions, due date 27th June and 10% weight in the final grade, for which the particular student has got 40%, and has thus not passed it. An example of information provided about estimated time can be seen in Fig. 9, and it distinguishes between time estimated for watching videos, going through readings and working on assessment each week.



	Due	Weight	Passed	Grade
Your First Computer Program				
Quiz: Quiz 1 10 questions Overdue	Jun 27	10%	✖	40%
Code Gone Loopy!				
Quiz: Quiz 2 10 questions Overdue	Jul 4	10%	–	–
Remixing Games				
Quiz: Quiz 3 10 questions Overdue	Jul 11	10%	–	–
Peer-graded Assignment: Project 1 2h Overdue	Jul 11	20%	–	–
Review Your Peers: Project 1 Overdue	Jul 14	20%	–	–
Reusing Your Code				
Quiz: Quiz 4 10 questions Overdue	Jul 18	10%	–	–

Figure 8: Personal progress (from "Code Yourself!")



WEEK 5 **Estimated time: 3h 31m**

Think Like a Software Engineer

Videos **1h 1m left**

Readings **10 min left**

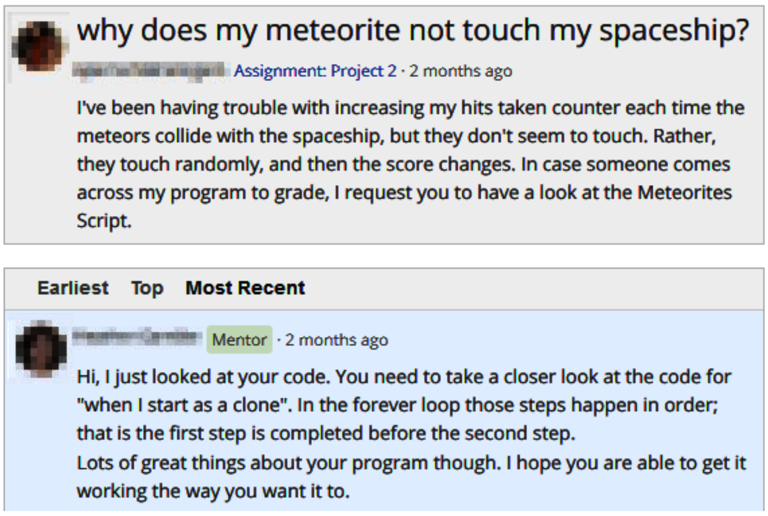
REQUIRED	GRADE	DUE
Quiz Quiz 5 **20 min**		Jul 24
Peer-graded Assignment Project 2 **2h**		Jul 24
Review Your Peers Project 2		Jul 27

Figure 9: Activities and estimated time (from "Code Yourself!")

Assessment policies were adjusted by Coursera to ensure that enough time was given for each assessment activity, as well as to allow for flexibility regarding submission dates. Students that have reached the course completion date, without having completed different assessment activities, are given the opportunity to switch to the next session. When switching sessions, students can keep most of their completed work, including course progress, quizzes, programming assignments that have been graded, and peer-reviewed assignments already submitted. Each quiz can be attempted up to 5 times per day. Peer evaluation is no longer anonymous, as information about the submission author and evaluators is provided. Information on the individual score given by each evaluator is also provided, while in the "fixed session" format session only the average grade was given. It is also possible to resubmit a project after its evaluation, so as to improve the project grade.

A major change decided by Coursera is that the free completion certificate is no longer available. Students can see through their personal account that they have completed a course, but they are not automatically provided with a certificate. Paid-for, verified certificates are still available, where the student identity is verified via webcam and typing pattern.

A positive development in course delivery that was brought by Coursera is the inclusion of "Community Mentors", who are volunteers who support learners by providing help and feedback. Community mentors answer student questions in the discussion forums, they post new threads to spark discussion and they provide feedback to Coursera staff. They can also interact with the course instructors via the private mentor forum. Both "¡A Programar!" and "Code Yourself!" were assigned community mentors. These were past students that were invited by Coursera for this role, as they had previously passed the course with a good grade and were active in the course by helping other learners in the forums. Initially, all community mentors were active in the forums, but, with time, interest from some of them dropped. We now have a small number of community mentors, who are, however, really active and make a considerable contribution to the learning experience. In "Code Yourself!", for instance, the two main community mentors have created more than 890 posts each and they have received around 200 upvotes in total. Fig. 10 presents an illustrative example of a mentor in "Code Yourself!" supporting a student in the forums. Note that the mentor does not readily provide a direct answer to the student's question, but instead gives the student hints so that they can discover the answer themselves. This is indicative of the pedagogical approach taken in our course, and the mentors successfully employ it.



why does my meteorite not touch my spaceship?

[redacted] Assignment: Project 2 · 2 months ago

I've been having trouble with increasing my hits taken counter each time the meteors collide with the spaceship, but they don't seem to touch. Rather, they touch randomly, and then the score changes. In case someone comes across my program to grade, I request you to have a look at the Meteorites Script.

[redacted] Mentor · 2 months ago

Hi, I just looked at your code. You need to take a closer look at the code for "when I start as a clone". In the forever loop those steps happen in order; that is the first step is completed before the second step.
Lots of great things about your program though. I hope you are able to get it working the way you want it to.

Figure 10: Example of mentor-student interaction in the discussion forums (from "Code Yourself!")

7 Results

In this section we present results for the two course modalities. The first modality corresponds to the session of March-April 2015, which was offered on the old Coursera platform in a "fixed session" format. The second modality involves the course delivery from August 2015 onwards in an auto-cohort mode on the new Coursera platform. As described in the previous section, the main differences between the two modalities involve assessments and scoring, minor corrections in quizzes, some new information resources, the Coursera website interface, student enrollment and flexible switching between sessions, absence of freely-available completion certificates, the ability to attempt quizzes and submit projects multiple times, peer evaluation no longer being anonymous and the introduction of community mentors. Results presented for the second modality involve the 14 sessions offered between August 2015 and September 2016. We present overall results for the two modalities, along with detailed information about demographics, the geographic distribution of our learners and results from course evaluation.

7.1 First session

As shown in Table 1, more than 59,500 people were enrolled in "Code Yourself!" and more than 25,200 in "¡A Programar!". The initial survey, which was not compulsory, was completed by around 10% of our learners in the English-speaking course and around 25% in the Spanish-speaking course. In relation to prior programming knowledge, over 82% of the participants indicated no or very little prior knowledge.

The proportion of participants that successfully completed the course was 2.68% in "Code Yourself!" and 6.30% in "¡A Programar!". Student retention (or completion rate) is defined as the fraction of individuals successfully finish a course (based on the standards specified by the instructor) over all students initially enrolled [59]. In a MOOC, high attrition is the norm [60]. For instance, completion rates of Coursera courses is approximately 5% [59], while Ho et al mention that average certification rates in Computer Science is 7% [61].

It is worth noting that retention should be considered in the context of learner intent, especially given the varied backgrounds and motivations [59]. As denoted by Zheng et al [30], there is a wide range of motivations for using a MOOC, and course completion and certification is only one of those. Many participants who enroll in MOOCs never have the intention to finish them at all. For instance, some people take a MOOC in order to gain insight into a particular area of study, but without deeming it necessary to view all lessons or complete all assignments [30]. It is worth mentioning that, according to student responses to the initial survey, the main motivations of our students were "to learn something new" (indicated by more than 92% of the students that completed the survey in both courses) and "to improve career options" (mentioned by more than 46%). Only 27% of "Code Yourself!" students and 37% of "¡A Programar!" students had the intention to pass the course.

In the initial survey, we asked whether students had enrolled on any MOOCs before. In "Code Yourself!" the percentage of those that answered it was the first MOOC was 58% and in "¡A Programar!" 57%. We also asked how they had heard about our MOOC (the students could select all options that apply). In "Code Yourself!" the main channels were the Coursera website (79%) and "friends/social networks" (17%). In "¡A Programar!" it was almost the same, with the percentages being 51% and 33%, respectively.

Table 1: Results from the first session

	First Session	
	Code Yourself!	***¡A Programar!***
Enrolled	59,531	25,255
Completed initial survey	6,229 (10.46%)	6,429 (25.45%)
Previous CS knowledge	82% no/little	89% no/little
Completed the course successfully	1,595	1,592
% completed/enrolled	2.68%	6.30%
This was the first MOOC ever attended	58% yes	57% yes
Main channels for finding out about the course	Coursera 79% Friends/social networks 17%	Coursera Website 51% Friends/social networks 33%

As far as engagement with the forums is concerned, both English- and Spanish-speaking courses included many lively discussions (see Table 2). During the first week of the course, in particular, there was a high level of enthusiasm in the forums, with people from all around the world introducing themselves and sharing their aspirations regarding the course. It is interesting to note that, even though enrollment numbers in "¡A Programar!" were less than half of those in "Code Yourself!", the number of students that contributed to discussions was higher (1,818 in contrast to 1,430 in "Code Yourself!"). The total number of threads and posts in the forums was also higher in "¡A Programar!" by approximately 40%.

Table 2: Student engagement with the discussion forums for the first session

	First session	
	Code Yourself!	***¡A Programar!***
Students contributed to discussions	1,430	1,818
Number of threads	776	1,081
Number of posts	4,204	5,940

Table 3 presents student demographic information for the first session. Regarding gender, the proportion of men and women was almost equal in the English version, and in the Spanish version it was two men for each woman. We consider this to be satisfactory gender inclusion, as according to Ho et al. [61], in CS courses there are four men for each woman, on average. The proportion of young learners in the first session was 15% in "A Programar" and 9% in

"Code Yourself". These are positive results compared to the 7.8% statistic reported by Nesterko et al [62] (including learners up to 20 years old), and the 2.85% indicated by Basogain et al [63].

It is also interesting that in the Spanish version almost all of the students who completed the initial survey are native Spanish speakers, while in the English version half of the students are English native. This difference could potentially explain why the proportion of students that completed the course surveys was higher in "¡A Programar!", as well as explain the difference in the course completion rates and the contributions in the forums. However, further research would be needed to verify this claim.

The geographical distribution of our students during the first course session is presented in Table 4. In "Code Yourself!" we had students from 197 countries, with high representation from North and Central America, Asia and Europe. In particular, a large proportion of the students were based in the United States and in India. In "¡A Programar!", we had greater representation from South America, followed by North and Central America. Countries with the highest participation were Mexico and Spain, which comes to no surprise, given that the course was delivered in Spanish.

Table 3: Student demographics for the first session

	First session	
	Code Yourself!	***¡A Programar!***
Gender	54% Male, 44% Female, 2% No answer	65% Male, 34% Female, 1% No answer
Youngsters (<=18 years old)	9%	15%
Is the language of the course your native language?	51% yes	97% yes

Table 4: Geographical distribution of students in the first session

	First session	
	Code Yourself!	***¡A Programar!***
Countries	197	117
Continents	North and Central America: 37%, Asia: 28%, Europe: 25%, Africa: 5%, South America: 4%, Rest of the world: 1%	South America: 40%, North and Central America: 35%, Europe: 23%, Asia:1,6%, Rest of the world: 0,4%
Top 5 countries	United States: 31%, India: 13%, United Kingdom: 4%, China: 4%, Canada: 3%	Mexico: 20%, Spain: 20%, Colombia: 12%, United States: 7%, Uruguay: 6%

We have collected feedback from the students through an optional survey at the end of the course, which was completed by 896 "Code Yourself!" and 1587 "¡A Programar!" students. Based on the survey, the course in both languages was positively evaluated across several criteria (see Table 5), including course length (more than 73% found it appropriate) and pacing (79% or more found it right). More importantly, at least 93% of the students found that the course covered or exceeded their expectations and at least 96% would recommend it to a friend. The course elements that were deemed more valuable were the videos, followed by quizzes and peer-assessed projects. An indication of the success of the course is that 90% or more of the students plan to continue programming in the future. It is also worth noting that 60-63% of the students indicated a commitment of 3-4 hours per week, which was the time commitment recommended by the course organizers.

In order to get an insight on engagement with assessments, we included a related question in the final survey. Based on feedback collected, 69% of "Code Yourself!" students and 66% of "¡A Programar!" students mentioned that they completed all assessments. In the case that they didn't, the main reason provided was that they "ran out of time" (29% and 26%, respectively). Other factors were "the certificate was not a priority" (10% and 8%, respectively) and "preferred to spend time on other elements of the course, e.g., lectures, forums" (13% and 6%). It is worth noting that the proportion of students indicated for completing all assessments might not fully represent the entire student population, given the course enrollment and completion numbers presented in Table 1. It is likely that many of the students that provided feedback were engaged throughout the entire course duration, and chose to complete more assessments than their classmates.

Table 5: Evaluation and student feedback for the first session (final survey)

	First session	
	Code Yourself!* *(896 surveys)	***¡A Programar!* *(1587 surveys)***
Course length	73% just right 21% slightly too short 3% slightly too long 3% very long / very short	78% just right 18% slightly too short 2% slightly too long 2% very long / very short
Course pacing	81% right	79% right
Met / exceeded expectations	93%	95%
Would recommend the course	96% certainly/probably	98% certainly/probably
Most valuable course elements (several options can be selected)	93% video lectures 54% quizzes 53% peer-assessed projects 42% resources 19% forums 11% community interaction	96% video lectures 65% quizzes 60% peer-assessed projects 53% resources 27% forums 18% community interaction
Plan to continue programming in the future	95%	90%
Commitment	<2 hours: 19% 2-3 hours: 34% 4-5 hours: 29% 6-7 hours: 8% more hours: 5% prefer not to say or unsure: 5%	<2 hours: 14% 2-3 hours: 35% 4-5 hours: 25% 6-7 hours: 9% more hours: 8% prefer not to say or unsure: 9%
Completed all assessments	69%	66%

7.2 Following sessions

This section presents results for the sessions offered between August 2015 and September 2016 in an auto-cohort mode. The total number of enrollments in the English version in this mode is significantly lower than that in the first session (Table 6), considering that it involves 14 sessions. The initial survey response rates are also lower compared to the first session (approximately 3% in "Code Yourself!" and 11% in "¡A Programar!"). A possible preliminary explanation could be the absence of weekly announcements issued by the course instructors, which, in the first session, highlighted the importance of feedback and reminded students about the survey. Regarding prior programming knowledge, 88% of the course participants in "Code Yourself!" and 92% in "¡A Programar!" indicated no or very little prior knowledge. This means that for the vast majority of the students, this course introduced a new area of knowledge, as in the first session.

Even though the motivations for registering for the course were similar, the proportion of students that indicated an intention to pass the course was somewhat lower than in the first session (20% in "Code Yourself!" and 31% in "¡A Programar!"). The completion rates were similar to those in the first session in the English-speaking course and somewhat lower for the Spanish-speaking course, but both were close to the typical completion rates for Coursera courses.

For the majority of our students, this was the first MOOC they ever attended (56% in "Code Yourself!" and 63% in "¡A Programar!"), which is similar to the case of the first session. For both courses, the main channels through which they found out about the course were the Coursera website ("Code Yourself!: 75%, "¡A Programar!": 46%) and social networks ("Code Yourself!": 15%, "¡A Programar!: 33%), although we did not promote the course with Google Ads this time. It is possible that social networking sites allowed for "word of mouth" promotion.

Table 6: Joint results for the following sessions

	Following sessions	
	Code Yourself!	***¡A Programar!***
Enrolled	40,105	37,025
Completed initial survey	1,186 (2.96%)	4,226 (11.41%)
Previous CS knowledge	88% no / little	92% no / little
Completed the course successfully	1,030	1,223
% completed/enrolled	2.57%	3.30%
This was the first MOOC ever attended	56% yes	63% yes
Main channels for finding out about the course	Coursera: 75% Friends/social networks : 15%	Coursera Website: 46% Friends/social networks : 33%

Discussion forums were less lively than in the first session (see Table 7). The number of students contributing to the “Code Yourself!” and “¡A Programar!” forums over the 14 sessions in the new modality is comparable to that for the first session only. This can be explained by the lower number of student enrollments. Similarly to the first session, the forum activity in “¡A Programar!” was more intensive that in “Code Yourself!”, with the number of threads and posts remaining higher.

Table 7: Student engagement with the discussion forums for the following sessions

	Following sessions	
	Code Yourself!	***¡A Programar!***
Students contributed to discussions	1,124	1,905
Number of threads	1,162	1,651
Number of posts	1,997	3,427

The gender distribution was maintained in “¡A Programar!”, while in “Code Yourself!” the participation of women was higher than for men (see Table 8), which we think is remarkable for a computing-related course. As far as young learner distribution is concerned, this was lower compared to the first session. It is still the case that about half of “Code Yourself!” students are not native English speakers, while the vast majority of “¡A Programar!” students are native Spanish speakers.

Table 8: Student demographics for the following sessions

	Following sessions	
	Code Yourself!	***¡A Programar!***
Gender	43% Male, 55% Female, 2% No answer	63% Male, 36% Female, 1% No answer
Youngsters (<=18 years old)	4%	7%
Is the language of the course your native language?	42% yes	98% yes

As can be seen on Table 9, the geographical distribution of our students was fairly similar that that during the first session.

Table 9: Geographical distribution of the following sessions

	Following sessions	
	Code Yourself!	***¡A Programar!***
Countries	194	116
Continents	North and Central America: 35%, Asia: 28%, Europe: 24%, Africa: 6%, South America: 5%, Rest of the world: 2%	South America: 43%, North and Central America: 38%, Europe: 17%, Asia:1%, Rest of the world: 1%
Top 5 countries	United States: 29%, India: 12%, United Kingdom: 5%, Canada: 4%, China: 3%	Mexico: 28%, Spain: 15%, Colombia: 13%, Peru: 7%, Argentina: 6%

We have circulated the same optional survey at the end of the course as for the first session, so as to compare results (see Table 10). The course length was still found appropriate (by 76% of the students or more), while the pacing was perceived by even more people to be right (89% as opposed to 79-81% in the first session). More importantly, the overall feedback that we got in the current sessions for both courses was even higher than in the first session, which was already very high. In particular, at least 97% of the students found that the course covered or exceeded their expectations and at least 98% would recommend it to a friend. A possible explanation could be the changes in the new modality, such as the flexible submission deadlines, the opportunity to switch to a subsequent session, and the availability of the course materials from the moment the session begins. It may also be related to differences in time commitment, which seems to have increased in the current sessions (see Tables 5 and 10). For instance, in the first session, 14-19% of the students stated that they dedicated less than 2 hours per week (with the suggested weekly commitment being 3-4 hours), while in the following sessions this percentage decreased to 5-8%. Further research would be needed, however, to draw concrete conclusions regarding the factors that lead to an improvement of student satisfaction with the course. It is also worth mentioning that an even higher proportion of the students plan to continue programming in the future (94-96%, as opposed to 90-95% in the first session).

The evaluation of different course elements has also improved overall (see Tables 5 and 10). In particular, there was a considerable increase in the student proportion that deemed course resources valuable (62-64% as opposed to 42-53% in the first session). Quizzes and peer-assessed projects also received a notable higher rating in the current sessions, which could be explained by the minor quiz corrections and the flexibility in submitting coursework. The discussion forums were the only course element that were found less valuable than in the first session. A possible explanation is that since August 2015, the average number of students per session is considerably lower than in the first session, thus leading to discussion forums that are, in general, more quiet.

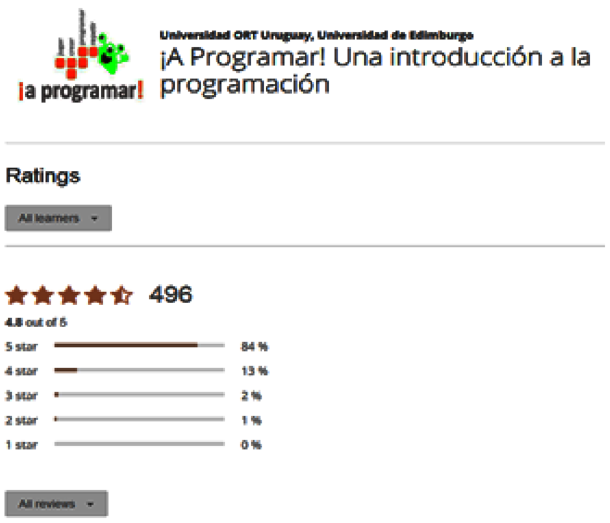
As far as assessments are concerned, 100% of “Code Yourself!” students and 99% of “¡A Programar!” students that participated in the survey indicated that they completed all assessments. In the case that they didn’t, the main reason provided was that they “ran out of time” (less than 1% in both cases). These results are better than in the first session, which could potentially be explained by the possibility to switch session. This should be checked with further research in the future.

Table 10: Evaluation and student feedback for the following sessions (final survey)

	Following Sessions	
	Code Yourself! (176 surveys)	***¡A Programar! (302 surveys)***
Course length	76% rigth 20% sligthly too short 3% slightly too long 1% very short/very long	88% rigth 8% slightly too short 3% slightly too long 1% very short/very long
Course pacing	89% right	89% right
Met / exceeded expectations	98.3%	97.7%
Would recommend the course	98.3 % certainly/probably	99 % certainly/probably
Most valuable course elements (several options can be selected)	98% video lectures 69% quizzes 61% peer assessed projects 64% resources 15% forums 11% community interaction	95% video lectures 71% quizzes 68% peer assessed projects 62 % resources 21% forums 19% community interaction
Plan to continue programming in the future	96%	94%
Commitment	<2 hours: 8% 2-3 hours: 48% 4-5 hours: 24% 6-7 hours: 10% more hours: 8% prefer not to say or unsure: 2%	<2 hours: 5% 2-3 hours: 31% 4-5 hours: 38% 6-7 hours: 12% more hours: 11% prefer not to say or unsure: 3%
Completed all assessments	100%	99%

In addition to feedback collected through our survey, we also got feedback collected by Coursera. Coursera has recently started prompting students to rate their courses, and the ratings are then included on the course webpage (see Fig. 11: Ratings and reviews). Each student may evaluate the course with one (bad) to five (excellent) stars and include a comment. In "Code Yourself!", 433 students rated the course and the average was 4.7, and "¡A Programar! got an average of 4.8 from 496 students.

The student reviews are publicly available. Most of them are short and rather general, like "very nice", "cute course", "great course", or "excellent". But, in some cases, more detailed information is included. In these cases, students mainly refer to organization, progression and, methodology: "Excellent methodology and content to learn concepts of software engineering and how to program. Recommendable for initiating in programming and even to those who already build their own programs", "It was just the right mix of how coding works accompanied by hands-on creating in a coding platform; "Well paced and organized". Other students highlighted that the course is a "starting point" and useful: "I really hope this would be my first step toward a career in computer science", "Put in the time to really understand the deeper usefulness of thinking like a programmer, and this course is very helpful. Scratch is a very fun way to learn about how to code, and gave me a nice foothold for beginning Java programming." Also, other students expressed how much they learnt through the course (e.g. "I learnt a lot", "I learned so much more that I bargained for and will be reviewing course content over and over"), or the use of Scratch (e.g. "Programming in Scratch can build solid foundations of IT knowledge", "Application to Scratch made that learning process really enjoyable"; and "Scratch is a cute tool for coding."). All these reviews may be useful for prospective as well as for existing students.


Universidad ORT Uruguay, Universidad de Edimburgo

¡A Programar! Una introducción a la programación

Ratings

All learners

496

4.8 out of 5

5 star	84 %
4 star	13 %
3 star	2 %
2 star	1 %
1 star	0 %

All reviews

Figure 11: Rating and reviews

7.3 Overall results and course team perceptions

Generally speaking, the course was very well received in both modalities, and it fulfilled the expectations of the participants, according to their survey responses. Overall ratings were slightly higher in the "flexible session" mode, which may be explained by improvements in terms of resources and assessment adjustments. Also, flexibility tunings (e.g. regarding coursework submission deadlines) could explain the improvement of opinion regarding the course pacing. It is also worth noting that initial conditions of the participants were similar, i.e. the majority indicated no prior programming knowledge. Furthermore, 90% or more indicated that they plan to continue programming in the future, which was one of our aspirations with this course: to provide beginners with a solid grounding in computing that would inspire them to develop their programming skills further in the future.

In order to complement students' perceptions, we conducted semi-structured interviews with staff members, mentors, teaching assistants and instructors. We asked their opinions related to their own experience and their perception of the course, including the main value of the course, its strengths and weaknesses, as well as differences between the two course delivery modes. A total number of seven "Code Yourself!" and four "¡A Programar!" stakeholders were interviewed. The interviews were recorded with the consent of the interviewees, transcribed and analyzed following thematic analysis. All stakeholders had a positive experience with the MOOC, and many expressed enthusiastic comments. For instance, the project coordinator for "¡A Programar!", who is a distance education analyst, highlighted that his experience with the MOOC was excellent, while the coordinator for "Code Yourself!" mentioned, among others, that "it's one of the big things in my life that I'm proud to be part of". One of the main themes that were highlighted by all members of the teaching teams is the fact that the course provides a gentle and accessible introduction to core programming principles. This was considered to be the main value of the course, while another point mentioned was the fact that the course makes the transition to other programming languages easier. Both course instructors emphasized the importance of outreach through this MOOC, and they found it highly motivating to develop teaching materials for a large, worldwide audience.

Another theme that was mentioned by senior members of the two teams is the importance of this international collaboration, and the challenges that it brought. Developing the course in two languages made the project bigger and more complex than traditional MOOCs, but, at the same time, it allowed the two teams to learn from each other, making the experience even richer. The bilingual production and delivery of the course was considered as the most remarkable point by three interviewees from the two teams. An important element of the course experience for teaching assistants and mentors was the fact that they kept learning by interacting with the students. One mentor, for instance, mentioned that "…(with the MOOC) I feel that I learned new things, it allowed me to exchange points of view with students and peers and to be part of a multidisciplinary team with a great sense of responsibility. I gained new knowledge and participated in a worldwide experience".

In general, interviewees had highly positive feedback and, thus, we had to insist so that they express their opinion on the weaknesses of the course. Two teaching assistants mentioned that, as with other MOOCs, some learners were confused about the peer review process. Two team members mentioned that the discussion forums in the auto-cohort mode tend to be more quiet compared to the first session. The "Code Yourself!" coordinator also mentioned that initially there were some challenges with delivering a Spanish-speaking course on the Coursera platform. Comparing the two course modes, some interviewees highlighted the flexibility that the auto-cohort mode offers, while others found that it now may be more challenging for the teaching team to commit time for running the course.

From an institutional point of view, the main motivation for this MOOC has to do with outreach and community engagement. As the Head of the School of Technology in Universidad ORT Uruguay stated, "it is an opportunity for widening participation, for democratizing the teaching of programming and for promoting interest in ICTs". According to the Principal of The University of Edinburgh, "We make our MOOCs for reputation, for fun, to try new ways of teaching, not for money. Online learning is an increasingly important method of teaching, opening up high-quality education opportunities to people around the world, we now see MOOCs as a powerful tool in our widening participation armory". Nevertheless, as for all institutions providing MOOCs, there are sustainability concern. As the MOOC design coordinator in the University of Edinburgh mentioned, "The one big thing with sustainability is that a lot of this stuff is harder to measure. If we don't make an effort to measure then obviously it's not going to be a priority because it is such a big investment."

To sum up, all the interviewees were highly motivated and with strong commitment, in all the steps for designing, producing and running the MOOC. They also agreed on the importance of the MOOC, from different points of view. Considering the results presented in Sections 7.1 and 7.2 and the opinions of staff members, mentors and teachers presented in this section, we believe that the objectives of the course have been achieved.

8 Discussion and Conclusions

In this work we presented the experience of Universidad ORT Uruguay and The University of Edinburgh in terms of the joint development and bilingual offering of a computer programming MOOC on Coursera. The course, called "Code Yourself!" in English and "¡A Programar!" in Spanish, aims to introduce youngsters to programming, while promoting the development of computational thinking and the use of basic software engineering practices.

The course was held simultaneously in an initial "fixed" session in March/April 2015 and then, it was adjusted to make it available in a "flexible" mode (it has been offered in this mode since August 2015). These changes include, among others, simultaneous sessions starting each month, the possibility for learners to switch to a subsequent session, and the availability of all course material as soon as a session begins. Similar results were observed in both modes and languages: high degree of satisfaction with the course and expression of interest to continue programming in the future, which is very positive, given that the vast majority of the students had no prior programming knowledge. Minor changes (for instance, regarding flexible submission dates) improved the course perception in terms of pacing and usefulness of different materials.

One of the limitations of this study has to do with the choice of student surveys for gauging student perceptions. Only a small proportion of students participated in the surveys (25% in the best case) and one could argue that respondents are most likely to exhibit a positive bias towards the course. Nevertheless, student evaluation is only part of the analysis presented in this paper. Student engagement, demographics and geographical distribution of students were also discussed. Following the literature recommendation to move beyond traditional approaches to evaluating college courses when evaluating MOOCs [58], our analysis includes not only completion rates, but also metrics around engagement with the discussion forums. In the future we wish to include additional metrics of student engagement, even though the research community does not seem to have agreed on such meaningful metrics.

It would also be interesting to further investigate engagement patterns, in particular with relation to different student motivations, such as "learning something new" or "getting a certificate". These are clearly different learner sub-populations, and it might be worth adapting the course design accordingly to fit their particular needs and aspirations. For instance, it is worth investigating whether including goal-based learning pathways leads to higher goal achievement and student satisfaction. Similarly, it would be useful to study whether different motivation-driven scaffolding strategies encourage engagement.

As future work, we also plan to further investigate the differences between the two modalities and their impact on student satisfaction with the course. We are considering both quantitative and qualitative methods for this task, such as interviews and focus groups, so as to get a better insight. Furthermore, we are working with the University of San Pablo to launch a Portuguese version of the course, called "Programe-se".

References

[1] B. Trilling, and C. Fadel, *21st century Skills, learning for life in our time*, Jossey-Bass, San Francisco, 2009

[2] ACM Computer Science Teachers Association. CSTA K–12 Computer Science Standards (2011) http://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Docs/Standards/CSTA_K-12_CSS.pdf. Accessed December 15, 2015. ISBN: 978-1-4503-0881-6

[3] A. Balanskat, and K. Engelhardt. (contributors), *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europa*. European Schoolnet, Belgium. October 2015. http://www.eun.org/documents/411753/817341/Computing+our+future_final_2015.pdf/d3780a64-1081-4488-8549-6033200e3c03

[4] MOOC: "¡A Programar!". Universidad ORT Uruguay and The University of Edinburgh. https://www.coursera.org/learn/a-programar. Accessed December 15, 2015.

[5] MOOC: "Code Yourself!". The University of Edinburgh and Universidad ORT Uruguay. https://www.coursera.org/learn/intro-programming. Accessed December 15, 2015.

[6] Framework for 21st Century Learning, http://www.p21.org/storage/documents/1.__p21_framework_2-pager.pdf. Accessed April 26, 2016.

[7] K. Falkner, R. Vivian, and N. Falkner, "Teaching Computational Thinking in K-6: The CSER Digital Technologies MOOC", *ACE 2015,* Sydney, Australia, 2015. ISBN: 978-1-921770-42-5. http://crpit.com/confpapers/CRPITV160Falkner.pdf

[8] I. Lee, F. Martin, and K. Apone (2011). Integrating Computational Thinking Across the K-8 Curriculum. ACM Inroads, Volume 5 Issue 4, p. 64-71, December 2014, DOI 10.1145/2684721.2684736

[9] J. Wing, Computational thinking. *Communications of the ACM*, Vol. 49, No. 3, March 2006. DOI 10.1145/1118178.1118215

[10] CSTA: Operational Definition of Computational Thinking for K-12 Education, https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf. Accessed March 14, 2016.

[11] A. Bundy, "Computational thinking is pervasive". *J. Sci. Pract. Comput.* 1, 67–69, 2007. https://core.ac.uk/download/pdf/28961399.pdf

[12] Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., and J. T. Korb. 2014. Computational thinking in elementary and secondary teacher education. *ACM Trans. Comput. Educ.*14, 1, Article 5, March 2014. DOI 10.1145/2576872.

[13] Mitchel Resnick, and David Siegel, "A Different Approach to Coding: How kids are making and remaking themselves from Scratch" https://medium.com/bright/a-different-approach-to-coding-d679b06d83a#.524r17diw. Accessed April 23, 2016.

[14] S. Grover, and R. Pea, "Computational Thinking in K–12. A Review of the State of the Field", *Educational Researcher*, vol. 42 no. 1 38-43, Jan/Feb 2013. DOI 10.3102/0013189X12463051.

[15] M. Zapata-Ros. "Pensamiento computacional: Una nueva alfabetizacion digital". *RED: Revista de Educación a Distancia,* 46(4), September 2015. DOI 10.6018/red/46/4.

[16] M. Sharples, A. Adams, N. Alozie, R. Ferguson, E, FitzGerald, M. Gaved, P. McAndrew, B. Means, J. Remold, B. Rienties, J. Roschelle, K. Vogt, D. Whitelock, and L. Yarnall, "Innovating Pedagogy 2015", Open University Innovation Report 4, Milton Keynes: The Open University, United Kingdom, 2015. http://oro.open.ac.uk/45319/

[17] M. Armoni, "Early Education – what does computing has to do with it and in what ways?", *WiPCSE'15 (Proc of the Workshop in Primary and Secondary Computing Education)*, ACM, USA, 2015. DOI 10.1145/2818314.2834898.

[18] L. Mannila, V. Dagiene, B. Demo, N. Grgurina, C. Mirolo, L. Rolandsson, and A. Settle,"Computational Thinking in K-9 Education", *ITiCSE-WGR'14*, Uppsala, Sweden, 2014. DOI 10.1145/2713609.2713610.

[19] Department for Education, "National curriculum in England: computing programmes of study (2013)". https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study. Accessed December 15, 2015.

[20] Code.org, https://code.org/ Accessed April 21, 2016.

[21] Codecademy, http://www.codecademy.com/. Accessed April 21, 2016.

[22] Scratch, http://wiki.scratch.mit.edu/wiki/Scratch Accessed December 15, 2015.

[23] Alice, http://www.alice.org. Accessed January 4, 2016.

[24] MIT AppInventor, http://appinventor.mit.edu/. Accessed December 15, 2015.

[25] S. Esper, S. Foster, W. Griswold, C. Herrera, and W. Snyder, "CodeSpells: Bridging Educational Language Features with Industry-standard Languages", *Koli Calling 2014*, Koli, Finland, 2014. DOI 10.1145/2674683.2674684.

[26] K. Brennan, and M. Resnick, "New Frameworks for Studying and Assessing the Development of Computational Thinking", *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada, 2012. https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf.

[27] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E.Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for All". *Communications of the ACM*, Vol. 52 No. 11, Pages 60-67, 2009. DOI 10.1145/1592761.1592779.

[28] Oxford Dictionary, http://www.oxforddictionaries.com/us/definition/american_english/MOOC. Accessed April 22, 2016.

[29] W. Siever, "Leveraging MOOCS", *Journal of Computing Sciences in Colleges.* Volume 30, Issue 2, Dec 2014. ISSN: 1937-4771

[30] S. Zheng, M. Rosson, P. Shih, and J. Carroll, "Understanding student motivation, behaviors, and perceptions in MOOCs", *CSCW 2015*, pp. 1882-1895, Canada, 2015. DOI 10.1145/2675133.2675217

[31] S. Cooper, and M. Sahami, "Reflections on Standford's MOOCs", *Communications of ACM*, V 56, N 2, February 2013. DOI 10.1145/2408776.2408787

[32] D. Malan, "Implementing a massive open online course (MOOC)", *Journal of Computing Sciences in Colleges*, Volume 28 Issue 6, p. 136-137, June 2013. ISSN:1937-4771

[33] Universities UK: "Massive open online courses. Higher education's digital moment?" http://www.universitiesuk.ac.uk/highereducation/Documents/2013/MassiveOpenOnlineCourses.pdf. Accessed December 16, 2015.

[34] A. Mc Auley, B. Stewart, G. Siemens, and D. Cormier, "The MOOC Model for Digital Practice". http://davecormier.com/edblog/wp-content/uploads/MOOC_Final.pdf. Accessed December 16, 2015.

[35] Coursera, https://www.coursera.org/. Accessed December 15, 2015.

[36] edX, https://www.edx.org/. Accessed December 16, 2015.

[37] Udacity, https://www.udacity.com/. Accessed March 20, 2016

[38] Miríada, https://miriadax.net/home. Accessed March 21, 2016.

[39] Edsurge, "Udacity, Coursera and edX Now Claim Over 24 Million Students", https://www.edsurge.com/news/2015-09-08-udacity-coursera-and-edx-now-claim-over-24-million-students Accessed April 28, 2016.

[40] K. Button, "10 Lessons learned from Moocs", Education Dive, http://www.educationdive.com/news/10-lessons-learned-from-moocs/306113/. Accessed January 4, 2016.

[41] G. Sharrock, "Making sense of the MOOCs debate". Journal of Higher Education Policy and Management, vol. 37, no. 5, pp. 597-609, 2015. DOI: 10.1080/1360080X.2015.1079399

[42] M. Kent, and R. Bennett, "What was all that about? Peak MOOC hype and post-MOOC legacies", *Massive Open Online Courses and Higher Education: What Went Right, What Went Wrong and Where to Next?*, 2017.

[43] F. M. Hollands, and D. Tirthali, "MOOCs: Expectations and Reality. Full Report", Center for Benefit-Cost Studies of Education, Teachers College, Columbia University, NY, 2014. https://oerknowledgecloud.org/sites/oerknowledgecloud.org/files/MOOCs_Expectations_and_Reality.pdf

[44] L. Schmid, K. Manturuk, I. Simpkins, M. Goldwasser, M., and K. E. Whitfield, "Fulfilling the promise: do MOOCs reach the educationally underserved?", *Educational Media International*, vol. 52, no. 2, pp. 116-128, 2015. DOI: 10.1080/09523987.2015.1053288

[45] S. A. Bass, "Simple Solutions to Complex Problems—MOOCs as a Panacea?". *The Journal of General Education*, vol. 63 no. 4, pp. 256-268, 2014. DOI: 10.5325/jgeneeduc.63.4.0256

[46] M. Petre, “MOOCs schmoocs: the education is in the dialogues”, *ACM Inroads*, vol. 4, no. 4, pp. 22-23, 2013. DOI: 10.1145/2537753.2537762

[47] C. Alario Hoyos, M. Pérez-Sanagustín, C. Delgado, and P. J. Muñoz-Merino, "Recommendations for the design and deployment of MOOCs: insights about the MOOC digital education of the future deployed in MiríadaX", *TEEM '14 Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality*, Pp 403-408, ACM New York, NY, USA, 2014. DOI: 10.1145/2669711.2669931

[48] P. J. Guo, J. Kim, and R. Rubin, “How video production affects student engagement: an empirical study of MOOC videos”, *L@S '14 Proceedings of the first ACM conference on Learning @ scale conference*, USA, 2014. DOI: 10.1145/2556325.2566239

[49] P. Adamopoulus, “What makes a great MOOC? An interdisciplinary analysis of student retention in online courses”. *34th International Conference on Information Systems*, Milan, 2013. http://people.stern.nyu.edu/padamopo/What%20makes%20a%20great%20MOOC.pdf

[50] MOOC: “C50X: Introduction to Computer Science”, Harvard University https://www.edx.org/course/introduction-computer-science-harvardx-cs50x. Accessed January 4, 2016.

[51] MOOC: “Intro to Computer Science”, https://www.udacity.com/course/intro-to-computer-science--cs101. Accessed January 4, 2016.

[52] M. Ben-Ari, “MOOCs on introductory programming: a travelogue”. *Acm Inroads*, 4(2), 58-61, 2013. DOI: 10.1145/2465085.2465102

[53] MOOC: “Computer Science 101”, https://www.coursera.org/course/cs101, Accessed April 15, 2016

[54] MOOC: “My CS: Computer Science for Beginners”. https://www.edx.org/course/mycs-computer-science-beginners-harveymuddx-cs001x# Accessed December 15, 2015.

[55] MOOC: “Programming in Scratch”. Harvey-Mudd College. https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-0. Accessed December 15, 2015.

[56] MOOC: “Pensamiento Computacional en la escuela (2da. edición)” https://miriadax.net/web/pensamiento-computacional-en-la-escuela-2ed. Accessed December 15, 2015.

[57] I. Kereki, and J. V. Paulós, “SM4T: Scratch MOOC for Teens - A pioneer pilot experience in Uruguay”, *FIE 2014 Frontiers in Education*, Spain, 2014. DOI 10.1109/FIE.2014.7044264

[58] M. Bali, “MOOC Pedagogy: Gleaning Good Practice from existing MOOCs”, *MERLOT J. of online Learning and Teaching*, V. 10, N.1, March 2014. http://jolt.merlot.org/vol10no1/bali_0314.pdf

[59] D. Koller, A. Ng, C. Do and Z. Chen, “Retention and intention in massive open online courses: In depth”. *Educause Review*, http://er.educause.edu/articles/2013/6/retention-and-intention-in-massive-open-online-courses-in-depth, 2013.

[60] D. Coetzee, A. Fox, M. Hearst, y B. Hartmann., “Should your MOOC Forum use a reputation System?”, *CSCW 2014*, p. 1176-1187, 2014. DOI: 10.1145/2531602.2531657

[61] A. Ho, I. Chuang, J. Reich, C. Austun Coleman, J. Whitehill, C. Northcutt, J. Williams, J. Hansen, G. Lopez, and R. Petersen, “HarvardX and MITx: Two Years of Open Online Courses Fall 2012-Summer 2014”. Available at SSRN: http://ssrn.com/abstract=2586847 or http://dx.doi.org/10.2139/ssrn.2586847. Accessed March 21, 2016

[62] S. Nesterko, D. Seaton, K. Kashin, Q. Han, J. Reich, J. Waldo, I. Chuang I., and A. Ho, Age Composition *HarvardX Insights*, 2014, http://harvardx.harvard.edu/harvardx-insights/age-composition. Accessed March 22, 2016.

[63] X. Basogain, M. Olabe, and J. Olave, “Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje”, *RED- Revista de Educación a Distancia*, 46(6), 2015. Accessed April 26, 2016. DOI 10.6018/red/46/6