Exactas Programa: llevando la programación a cada rincón de la ciencia

Matías Lopez-Rosenfeld, Esteban Mocskos, Mariano González Lebrero, José Crespo, Mehrnoosh Arrar, Inés Caridi, and Mariela Sued

Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina
exactasprograma@dc.uba.ar

Resumen Las herramientas computacionales juegan un rol cada vez más importante en casi todas las disciplinas científicas y tecnológicas, así como el sector gubernamental y la industria. El saber programar, por otro lado, no ha sido formalmente incorporado como conocimiento requerido para los graduados de las distintas instituciones educativas, tales como la Facultad de Ciencias Exactas y Naturales (o simplemente *Exactas* como la llamamos) de la Universidad de Buenos Aires en Argentina. Esta fuerte contradicción podría llevarnos a que los futuros científicos tengan una pobre preparación en términos de las herramientas computacionales que deberán utilizar en sus tareas diarias.

Para solucionar este complejo escenario, es necesario incorporar el potencial que ofrece la Computación en las diferentes carreras. Saber cómo escribir un programa para resolver un problema es mucho que aprender a programar, es una aproximación activa que ayuda a los estudiantes a organizar su razonamiento lógico en forma de pasos claros y concisos. Ayuda a consolidar el entendimiento del problema en sí, más allá de la disciplina de que se trate.

Nuestro objetivo al crear *Exactas Programa* es proveer los elementos esenciales de programación, sin introducir una asignatura completa en su plan de estudios ya exigente, buscando que los estudiantes puedan incorporar a la computadora como una herramienta práctica que los asista en la resolución de problemas.

Con este objetivo en mente, hemos formado un equipo de trabajo conformado por profesores, auxiliares, estudiantes de doctorado de diferentes disciplinas para asegurar una mirada interdisciplinaria que combine diferentes motivaciones para el uso de la computadora para presentar desafíos transversales a las distintas disciplinas. El resultado, *Exactas Programa*, no es un curso de programación, es un taller de resolución de problemas en el que la computadora tiene el rol principal.

En este trabajo, compartimos la estructura del curso, los detalles de las actividades incluidas y algunas lecciones que aprendimos luego de haber dictado el taller en cinco oportunidades.

Keywords: Enseñanza Programación · Ciencia · Actividades Lúdicas.

Received December 2020; Accepted March 2021; Published June 2021

School of Sciences embraces programming (*Exactas Programa*): reaching every corner of Science

Matías Lopez-Rosenfeld, Esteban Mocskos, Mariano González Lebrero, José Crespo, Mehrnoosh Arrar, Inés Caridi, and Mariela Sued

Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina
exactasprograma@dc.uba.ar

Abstract. Computational tools play an increasingly central role in almost all scientific and technological disciplines, as well as throughout both Government and Industry sectors. Programming skills, on the contrary, have not been formally incorporated as required knowledge for graduates of educational institutions, such as the School of Sciences (or simply *Exactas* as we call it in Spanish) of the Universidad de Buenos Aires in Argentina. This sharp contradiction could lead to the poor preparation of future scientists in terms of the necessary use of computational tools in their daily activities.
To address this skill gap, it is necessary to incorporate the great potential of computing into the different curricula. Knowing how to write a program to solve a problem is far more than learning to write a piece of code; it is an active learning approach that helps students organize the logical reasoning steps and fosters a solid understanding of the subject matter, regardless of the discipline.
Our objective in creating *Exactas Programs* was to provide the essential elements of programming- without introducing another stumbling block in their already challenging degree programs- so that students of any major can incorporate the computer as a practical problem solving tool. With this objective in mind, we formed a working group of professors, teaching assistants, and doctoral students of different scientific backgrounds to ensure a multidisciplinary initiative that combines different motivations and contexts for the use of the computer in addressing challenges that traverse all majors. The result, *Exactas Programs*, is not a programming course; it is a short problem-solving workshop in which the computer is the central tool.
In this work, we share our proposal's structure, the details of some of the activities that comprise it, and the lessons learned after five editions of the workshop.

Keywords: Programming Teaching · Science· Gaming Activities.

1 Introduction

The use of the computer as a tool has permeated all fields of science and engineering. Today, it has become one of the pillars that sustains development and

innovation [11]. Programming also constitutes an invaluable tool in the learning process in almost every area of knowledge. A person capable of working with algorithms has training that prepares her for much more than writing useful programs. She has a general-purpose instrument that will be determinant in her/his development.

It is often said that a subject is truly understood when it is possible to teach it to someone else. Developing a program is a way of teaching something to a computer. Writing a program requires a deep understanding of the problem as a critical step. Furthermore, during the development process itself, the programmer is forced to arrange and clarify the necessary steps to obtain a solution, which helps organize the thought process.

In the case of a person dedicated to scientific activity, the ability to program turns out to be a highly valued skill. Even in various areas of knowledge, it is essential to perform well at a professional level. Motivated by the great potential of the computer as a problem-solving tool, we created a working group of teachers, researchers, and doctoral students from different departments of the School of Sciences of the University of Buenos Aires (FCEN-UBA) and designed a pilot experience to ensure members of our community learn to solve problems using a programming language.

In the middle term, our goal is that this workshop favor the incorporation of computational practices within the content of the different degree programs of FCEN-UBA. We seek to make evident that learning how to program is an accessible tool to all students and that its use is transversal to all disciplines.

Learning to program is, in general, a hard task for many students. This fact has been widely reported in literature [12] and testified by the high dropout rates in first-year courses [3,7]. Despite the recent coding craze and boom in the tech industry, a clear understanding of how programming is actually learned by students is still lacking. One phenonomenographic research effort [4] is categorize ways in which students experience the act of learning to program in a first-year University programming unit, highlighting a large variation in how students' perceive their introductory programming learning experience. One of the more sophisticated of these learning experiences is centered around problem solving, which is also the core of an integrated approach to teach programming and problem solving called Computational Thinking [10]. Many research efforts have honed in on how the choice of programming language itself and incorporation of different tools can enhance introductory programming learning experiences. In particular, Mannila et al. [9] defend the use of Python as an initiatory language and Ambikesh Jayal et al. [8] show that starting with Python (and then moving to Java) have better results than beginning directly with Java. We can also mention works that show IDEs or libraries also help in introductory programming units [6,2]. Among the many examples of teaching programming for future scientists, we also highlight the interesting experience of Arms et al. [1] who use Jupyter notebooks to teach coding to atmospheric science students with great results.

In 2015, the Federal Council of Education of the Argentine Republic declared teaching and learning programming techniques of strategic importance for the system. Argentine education during compulsory schooling to strengthen the economic and social development of our country [5]. But this norm does not reach the university environment. Only a few universities include programming-related content in their curricula, and most of the first-year students have never had a programming class before. We put forward a proposal that supports an approach to resolving different types of situations using a computer so that any member of the FCEN-UBA can carry it out.

This paper describes this project, called *Exactas Programa*, a problem-solving workshop developed with the perspective of different scientific disciplines using the computer as the central tool. Next, we explain the objectives and how we implemented the proposed activities.

2 Our proposal

The founding objective of *Exactas Programa* is to incorporate programming into the training of our students from the very beginning of their degree programs. We seek to present computational tools that allow students to solve different problems, promote algorithmic reasoning, and implement their codes in a programming language. This approach will later facilitate the inclusion of new practices based on programming within the subjects taught nowadays at our University (FCEN-UBA).

We designed a workshop that consists of nine meetings of four hours each, with classes three times a week. To date, we delivered five editions during the summer and winter breaks, based on the pilot experience launched in the summer of 2018. It is important to note that the course does not award credit hours or points to students, nor is it compulsory, so public attendance is motivated exclusively by interest in the subject.

The course is not an introduction to programming but presents basic programming notions necessary to solve different challenging situations. The modality of the classes is theoretical-practical. They take place in a computer laboratory where some of the students use the computers already installed whereas others use their personal laptops. In any case, one of the critical conditions is that each participant uses a computer and performs the activities individually, in such a way that the learning is active and that each attendee writes their own code.

During the development of the activities, the students receive personalized assistance from the teaching team to transform their solution ideas into a code that can answer the questions. To assist and accompany the participants' needs, we have one teacher for every eight students; an unusual relationship in areas such as ours but has been essential to navigate the rocky path proposed by the workshop successfully. The teacher-student relationship is key to ensuring that a student without prior programming knowledge can write relatively complex programs already in the second class.

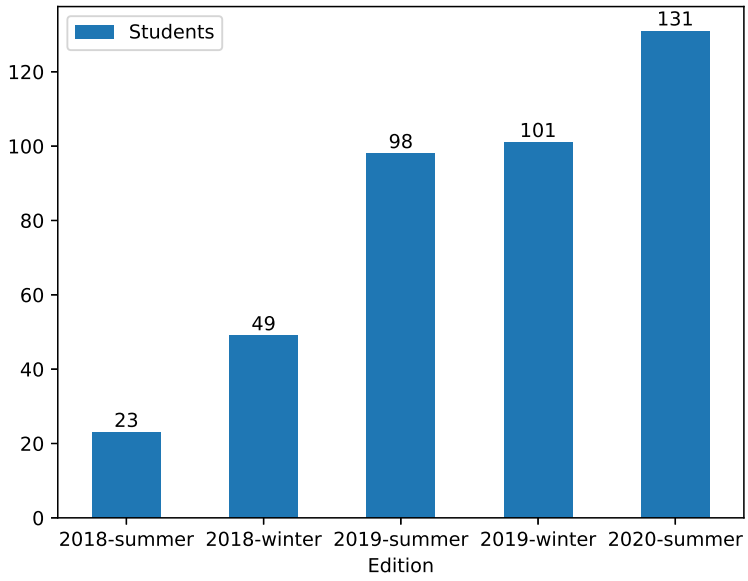

Fig. 1. Number of students who completed the activity in the different editions of the workshop. This proposal workshop was designed to be delivered during the winter and summer academic recesses.

Throughout the nine meetings, the workshop proposes seven activities in the form of problems/challenges:

i) Play Black Jack.
ii) Simulate filling a stickers album.
iii) Model the forest fire dynamics.
iv) Model and visualize the orbit of the earth around the Sun.
v) Simulate the evolution of an avalanche.
vi) Model the predator-prey interaction in time and space.
vii) Simulate the dynamics of the interaction of particles in two dimensions.

An initial motivation presents each of these activities that later give rise to a series of questions. After a brief presentation, the need for a program to help us answer the questions of interest becomes clear. For example, what is the probability of filling the World Cup album by buying 80 packages of stickers? [1]

The seven activities mentioned are formulated in an attractive way, including, in each case, a hands-on activity that allows students to approach the problem tangibly, either by playing or experimenting with different unplugged materials:

[1] This question is usually addressed every four years in the media: Mundial 2014 https://www.pagina12.com.ar/diario/contratapa/13-250187-2014-07-06.html; Mundial 2018 https://www.lanacion.com.ar/2125275-rusia-2018-cuantos-sobres-de-figuritas-hacen-falta-para-llenar-el-album-del-mundial

papers, balls, letters, dice, candy, etc. The possibility of experimenting and performing live gives strong support to develop the algorithms later and write the programs that can carry out the proposed instructions, following and solving a carefully elaborated practical guide for the rest of the class. The active part of each class helps to clarify the critical question of interest and the dynamics of the system to be studied. In addition to serving as a way to understand the details of the problem at hand, the hands-on activity also helps create a friendly bond in the group, improving the study climate and the exchange of ideas. Finally, it helps so that the teachers, during the consultations, can refer back to specific moments during the doubts that arise when coding (for example, *"Do you remember that when we played ...?"*).

Table 1 summarizes the general plan for most classes. The meetings have a series of programmed activities that allow to present the activity and understand the system's details to study and program. Encounters that do not follow this structure are the first class and the last. In the case of the first one, it provides the necessary programming concepts so that a student without prior knowledge can write their early programs. The last class is also different since it includes a group reflection and discussion on the workshop itself.

Stages	**Expected product**	**Available materials**	**Time**
Close previous activity class	Highlight typical situations and frequent mistakes	Blackboard and slides	30 min
Presentation of the problem	Recognize the objects involved	Slides	10 min
Posing questions		Blackboard and slides	20 min
Experimentation *hands-on activity*	Understanding the dynamics involved in the problem	Dices, pencil, paper, candies, etc.	15 min
Closure of experimentation	Identification of how to model the objects involved	Blackboard	10 min
Implementation	Program that allows you to answer the questions	Exercise guide + computers + teachers	2.5 hs

Table 1. The general structure of the workshop classes. The meetings have a series of scheduled activities that present the problem, understand the system's details to study, and implement the code to solve the problem at hand.

The teaching team has an interdisciplinary composition that provides a great wealth of approaches and shows the different ways in which each discipline can use programming. The workshop shows that programming is not an exclusive tool for computer specialists but offers immense possibilities to all areas. This richness in the variety of the teaching team's training allows everyone to feel mainly motivated by any of the proposed activities related to cross-sectional knowledge of the School of Science's degree programs.

The didactic proposal of *Exactas Programa* is based on the use of the `Python` programming language, as it is an open-source language that is increasingly used

both in the scientific community and in the industry. However, since the course's focus is on using the computer to solve problems and not on technical details of the language, advanced concepts typical of `Python` (such as comprehension lists, iterators, etc.) are omitted. In this way, the skills acquired in the course are pretty applicable to others (`R`, `Matlab`, `Fortan`, etc).

3 Activities

The first meeting of the course presents `Python` and briefly explains its syntax, the concepts of assignment, loops, conditionals, functions, and some basic notions of data types (`int`, `float`, `string`) and lists. From the beginning, it seeks to complement the explained concepts with tests carried out by the students on the computer.

To avoid problems with the installation of tools and delays, in the beginning, we opted for the use of a web environment that offers the possibility of writing and testing Python programs in a very intuitive way. `Python Tutor`[2] is a web service that allows you to write, execute and analyze the behavior of programs in `Python` without the need to have any tools installed. Students only need a browser and Internet connection, both available in the FCEN-UBA computer labs.

During the course, students can solve the activities primarily by combining the tools seen in the first class, except for matrices (with `numpy` package) and graphics (with `matplotlib`) plus some modules detailed in further activities. It is important to note that a great emphasis is placed on thinking *what* each variable and list really represents throughout the course. We proposed a strong value to the semantics with which variables or lists are interpreted (for example, a list of zeros represents an empty stickers album, and that will be the way to call it, see more detail of the album in the section 3.2).

This workshop was conceived for students without any programming experience. The first classes represent a significant challenge in transmitting the fundamental programming concepts to solve exciting problems immediately. Then, the remaining meetings deepened concepts and proposed more complex challenges. After a few encounters, the students can visualize that they can recreate complex scenarios and ideas with just a few elements.

Two of the activities proposed throughout the course are detailed below. For each one of them, we describe the model proposed, the open questions to answer with this model, the hands-on activities, and the programming concepts necessary to complete it.

3.1 NanoJack

This activity simulates a Black-Jack card game's simplification: the game consists of each player receiving cards whose values are between 1 and 13 in a

[2] www.pythontutor.com

continuing way while he/she add their values. When the player reaches exactly the value 21, he/she wins but, if this value is surpassed, she/he loses. After recreating the dynamics of the single-player game, the number of players increases. The question to be answered is what proportion of players win this game.

Before starting to program, in the context of the gaming section, groups of four or five students join a teacher who acts as a dealer, and they play NanoJack live. The necessary material is one deck of French-suited playing cards per group.

Although each group first defines how they will play the game, one of the options is that the dealer deals cards one by one to the first student of the round until the cut-off condition is met (reaching or exceeding the value 21), following with the other students. As the game progresses, the teacher invites the students to relate what is happening. These ideas, expressed in words, will then guide the implementation.

This initial sharing is of a very high level and with little detail on writing a piece of software. Still, it allows students to visualize ideas and necessary model elements on a concrete substrate and then be able to take them to the abstraction of the code. Also, during this time, the teachers introduce themselves to the students, helping to create a nice working environment.

In this activity, students should exercise lists (dynamic creation and traversal), nested loops with counters (in creating a four-suit deck), and loops with Boolean conditions (sum reaches or exceeds 21), as well as usage from the `random` module to use the `shuffle` function. Also, they are already beginning to experiment with testing to know if their program does what they expect. Thus, students early understand that debugging a code is an essential task that any programmer constantly faces.

3.2 Stickers

Filling a sticker album is an activity that the vast majority of students and teachers, if not all, claim to have done at some point in their lives. Throughout this meeting, we propose a series of questions about this problem and try to answer them through computer simulation. Although mathematics has widely approached this subject, we only mention a few references in this regard, without venturing into this path to solve the challenges posed.

We can summarize the activity of collecting stickers in the following way: 1) we acquire an album (empty book with numbered places to be completed); 2) we buy packs of stickers one by one. Each pack contains numbered stickers that must be pasted in the album in the site corresponding to its number; 3) we have won when the album is completed.

The fact that the stickers (in particular, those of the World Cup Sticker Album) are so widespread in Argentina helps since the task does not require complex explanations. Moreover, it usually brings back some memories of the students from their own experiences: the existence of *the difficult ones*, the composition of the packages, and many questions that arise spontaneously from the class.

After a brief discussion, we agreed to answer the following questions: on average, how many packages of stickers must be purchased to fill the album? What is the probability of filling the album if I can buy 700 packages at most?

Figure 2 shows the simplified album model used during the hands-on activity. An album with a capacity for six stickers is given out, and we emulate the chance of buying one sticker by rolling a dice. The students must calculate how many *stickers* they had to buy to fill the album and then obtain the average of the amount needed. The didactic material consists of a paper album with six sticker slots to be filled and a die for each group of two or three students.

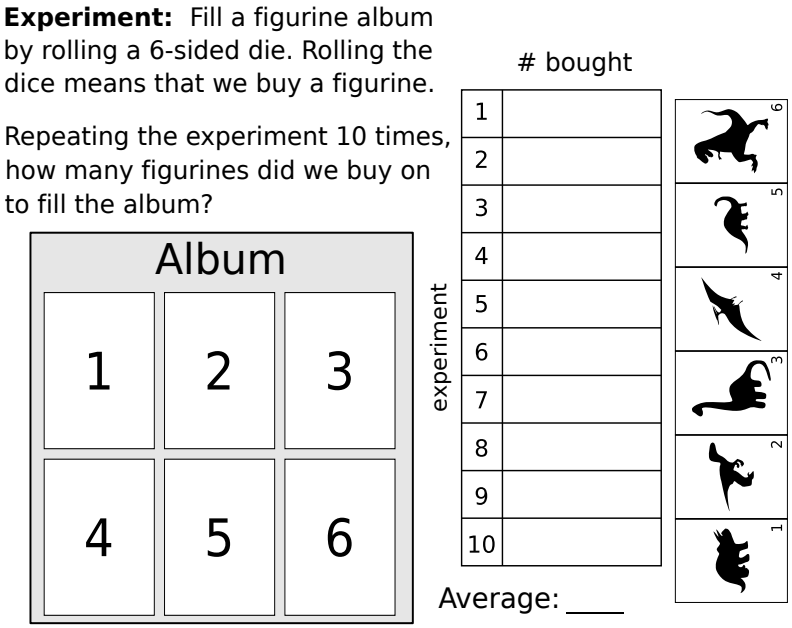

Fig. 2. Images of the album and stickers used in the hands-on activity of the second class. Students can perform the mechanics of filling an album and understand the main concepts related to that process.

To reach the final model (a large album and packages of stickers) a first implementation is made in which the *stickers* are bought one at a time (following the logic of the hands-on activity), and then it is extended to stickers obtained in packages of five.

Together, these NanoJack and Stickers activities represent the core of the course's logic and where students incorporate the greatest number of tools.

4 Results and Discussion

At the beginning of the pilot workshop in 2018, our ambitious long-term goal was that all the incoming students of Exactas attend the workshop. After these first five editions, we continue to find an interest that seems insatiable, both from our university and from other institutions. We continue on the path of ensuring that this workshop is of interest and made available to students of any degree and that it is helpful to them. Below we detail some aspects of the workshop that we worked on during its first five editions to get closer to this objective.

Multidisciplinary teaching staff The *Exactas programa* workshop was conceived by a multidisciplinary teaching team to provide a powerful tool to students of any degree program in our university. The workshop activities are inspired by exciting problems from these different disciplines but always focus on computational challenges. Throughout the different editions, we have observed the need to maintain this diversity in the teaching staff but with a core of teachers who remain stable during the nine meetings. The students find their referents in the first classes, they choose with whom they understand each other better by asking and resolving doubts, and these relationships are sustained and strengthened throughout the workshop.

With this multidisciplinary vision and, by increasing the number of simultaneous shifts incorporated, it was necessary to maintain this spirit and, at the same time, ensure that there was a solid technical reference (computational) in each shift. It is usual for students to raise doubts or concerns that may exceed the workshop's concepts or that an error occurs that may be complex to solve. The presence of a technical reference with solid knowledge of `Python` in each shift proved to be more than enough to be able to answer or solve any technical problem that arose.

Hands-on activities The fact that each challenge begins with a hands-on activity and later sharing in words or on the blackboard the ideas of pseudocode allows, on the one hand, to understand the problem and, on the other, it helps to train computational-algorithmic thinking before any attempting any implementation. These fun hands-on activities require identifying which objects make up the model and a division of the tasks that necessarily and/or strategically allow to break down the dynamics of the system into its basic rules.

4.1 Balance between creativity and structure

The multi-disciplinary team teaching approach has helped not only in identifying problems of diverse interest, but also in seeing the diversity in possible ways of thinking through and solving the problems we propose. One of the first decisions we were faced with in designing the course was how much to guide or bias the way in which the problem was solved; on the one hand to not lose students who

did not know how to begin and on the other to not confuse others who would have constructed their solutions differently.

As an example, we return to the sticker album activity. In the hands-on activity, the students quickly understand the basic rules of the game: we buy a sticker and we locate it in the album, over and over again until the album has been filled. We observe, nevertheless, a great deal of diversity in the ways in which the students model this dynamic, which branches at four key decisions: how to randomly generate a sticker, how to represent the empty album, how to locate the sticker in the album, and how to define the conditional that determines whether or not to continue iterations.

In the first edition of *Exactas Programa*, we attempted to avoid biasing the solutions of the students, encouraging full creativity, and only providing some potentially helpful commands, such as `random.random()` or `random.randint(a,b)` and a prompt such as "Simulate how an album of six stickers is filled".

Based on student feedback and intending to reduce the dropout rate after this activity (the second activity of the workshop), we began to further structure the problem's resolution. For example, we added exercises like " Define a function that takes a list `album` as an input and a `figu` element as a parameter, and returns the `True` or `False` output indicating whether or not `figu` is in the `album` list.

In the same way, in the sharing after the hands-on activity and before starting to program, we discuss different ways of facing the problem. We unify on the blackboard a way to solve the problem (in a pseudo-code), making explicit (for example, that the empty album will be a list of zeros and the whole album will be a list of ones). In each edition of *Exactas Programa*, we usually adjust exercises in the guides to continue looking for this balance between greater structure (less frustration) and less structure (greater creativity).

4.2 Scalability and portability

In its first edition, *Exactas programa* was a workshop given by teachers and doctoral students from our Faculty. Each teacher was in charge of developing, and then dictating, an activity related to her discipline. Due to our community's broad interest in attending the workshop, and thanks to our Faculty's support, we have multiplied in size, doubling the number of shifts in the subsequent two editions. During this stage, the need for general coordination of the workshop became evident. Thus, we developed a web page and a registration form. Also, the allocation of places in the four shifts, the teaching staff's distribution, and the appointment of a coordinator per shift helped develop the workshop satisfactorily.

Today we have a repository with all the course material, to which all teachers have access. The repository allows a unification of the course material, such as changes to the exercises and class slides that introduce each activity. To homogenize the resolution of the problems and ensure that a teacher from any area can guide each of the activities, we greatly benefited from creating a teaching guide for each activity. We seek to adjust the course material to continue increasing

the number of workshop quotas and help its export to other institutions, such as the National University of San Martín, Argentina, in the Summer of 2020.

4.3 Feedback from various perspectives

We believe that a workshop with a broad audience and teachers from so many different areas requires an almost continuous search for feedback. At the end of each meeting, attendees must complete a form that allows us to know their impressions of the proposed activity, get feedback on aspects such as how much of the activity they were able to finish during the class time, how difficult it seemed, etc. Also, each student must submit the code that she/he produced during class. This material allows us to study the different routes taken in each participant's learning process and helps the teacher construct a closing of the activity for the following class. In this same sense, a time is always reserved for the last meeting to discuss with the students about suggestions and general impressions of the workshop. We also sent a final survey to all attendees pursuing the same objective.

For each activity we ask the students to grade their agreement to the statements: Was the activity useful?, and Was the activity entertaining? The values correspond to the following categories:

- -2: "completely disagree"
- -1: "partially disagree"
- 0: "neutral"
- 1: "partially agree"
- 2: "completely agree"

Figure 3 presents a view of the end-of-workshop survey results combining Usefulness and Entertainment dimensions. In all the activities most of the students found them both useful and engaging during the complete edition. This results in an interesting feedback and highlights that the workshop composition (students with low or no previous experience in programming) help to keep the group interested and challenged by our proposals.

On the other hand, we have benefited from a teaching meeting after the end of the workshop to discuss general results and impressions of the various shifts and ideas to improve the course. Each teacher also completes forms to propose specific changes to the workshop activities. As a result of incorporating new teachers, we have prepared three essential guides where we summarize the main concepts addressed in the first two activities and propose some exercises related to them.

5 Conclusions

Exactas Programa is a sensory, face-to-face experience. There is music, dramatization, cakes that teachers and students bring. We put into practice fundamental coexistence laws promoted, among others, by Greg Wilson (see Rules section of

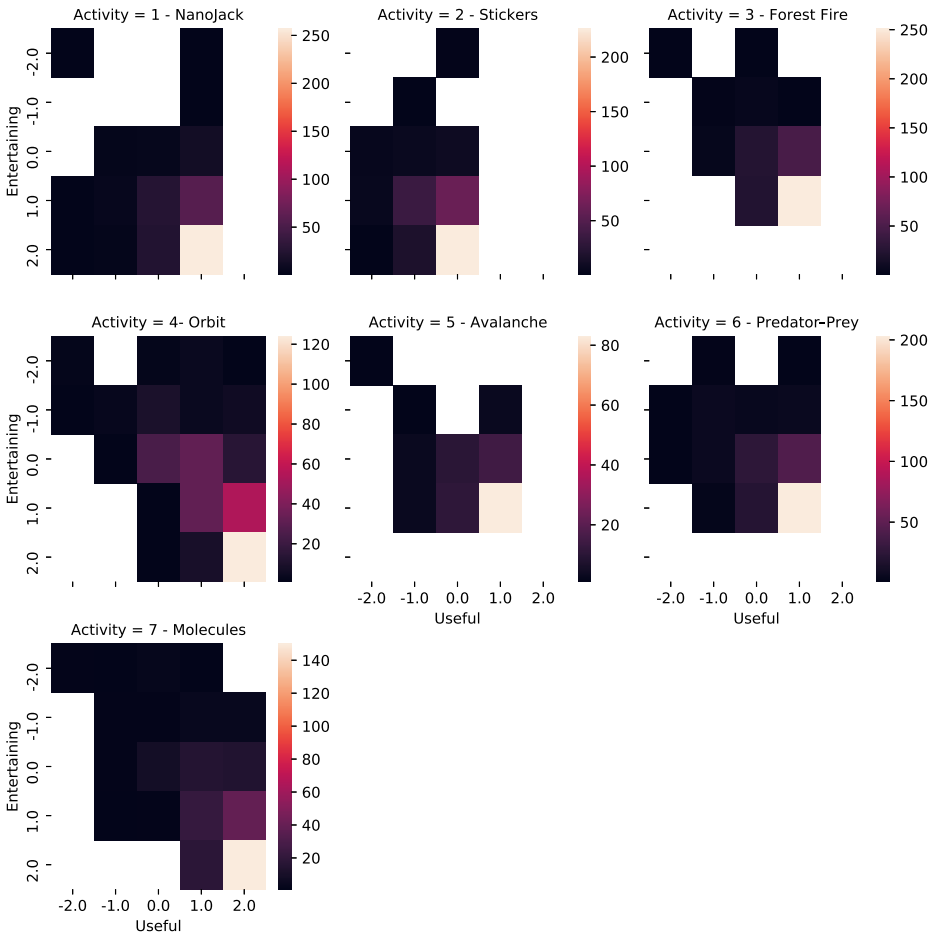

Fig. 3. Survey Results from 2019 edition including Usefulness and Entertainment dimensions. Color codes the number of students who answered each combination (brighter means more students).

his book [13]). In this way, we create an environment where empathy becomes the fundamental tool for sharing and generating knowledge. We generate communication channels (a WhatsApp group per turn) where the attendees raise their doubts and, generally, a peer manages to respond. Teachers are also involved in this communication space.

This space is important both because of how dynamic it is to assist students in minor complications and because of the collaborative network established between the participants, including the teachers.

Finally, we want to mention that we have managed to reduce desertion throughout the different editions, which occurred mainly after the first or second meeting. We have taken several measures: reformulation of the classes, intensification of the nearly insistent accompaniment, extra ad-hoc introductory guides summarizing the main concepts. We have not studied the impact of each of these measures, but together, they have effectively retained students.

6 Final Considerations

During the last ten years, there has been an explosion in the Computer Education field: published works, doctorates, and, in some cases, many financial resources accompanying different initiatives. Nevertheless, the results have not been as expected, and specialists in the area have been critically reflecting on the matter, promoting new practices.

Our initial proposal, conceived by a teaching staff with many years of experience working in our Faculty but little theoretical training in the field, includes the goals:

- Integrating computing with other disciplines, presenting a wide range of problems to solve.
- Proposing activities that require active participation by all attendees.
- Not seeking efficiency in the resolutions, teachers listen and follow students' trajectories, trying just to clarify serious conceptual errors.
- Promoting computational thinking beyond implementation when thinking about new challenges.

We hope that our experience can help those who want to promote this type of initiative.

Acknowledge Special thanks to those teachers, collaborators, and authorities who made *Exactas Programa* possible and all our gratitude to those students who participated in this experience.

References

1. Arms, S., Chastang, J., Grover, M., Thielen, J., Wilson, M., Dirks, D.: Introducing students to scientific python for atmospheric science. Bulletin of the American Meteorological Society **101**(9), E1492 – E1496 (Sep 2020). https://doi.org/10.1175/BAMS-D-20-0069.1

2. Begosso, L.C., Begosso, L.R., Gonçalves, E.M., Gonçalves, J.R.: An approach for teaching algorithms and computer programming using greenfoot and python. In: 2012 Frontiers in Education Conference Proceedings. pp. 1–6 (2012). https://doi.org/10.1109/FIE.2012.6462307
3. Blesa, M.J., Duch, A., Gabarró, J., Petit, J., Serna, M.: Continuous assessment in the evolution of a cs1 course: The pass rate/workload ratio. In: International Conference on Computer Supported Education. vol. 583, pp. 313–332. Springer (2015). https://doi.org/10.1007/978-3-319-29585-5_18
4. Bruce, C., Buckingham, L., Hynd, J., McMahon, C., Roggenkamp, M., Stoodley, I.: Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university. Journal of Information Technology Education: Research **3**(1), 145–160 (Jan 2004). https://doi.org/10.28945/294
5. Consejo Federal de Educación: Resolución Nro. 263/15 (2015), `https://cfe.educacion.gob.ar/resoluciones/res15/263-15_01.pdf`, visitada el June 15, 2021
6. Edwards, S.H., Tilden, D.S., Allevato, A.: Pythy: Improving the introductory python programming experience. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education. p. 641–646. SIGCSE '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2538862.2538977
7. Giannakos, M.N., Pappas, I.O., Jaccheri, L., Sampson, D.G.: Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. Education and Information Technologies **22**(5), 2365–2382 (Sep 2017). https://doi.org/10.1007/s10639-016-9538-1
8. Jayal, A., Lauria, S., Tucker, A., Swift, S.: Python for teaching introductory programming: A quantitative evaluation. Innovation in Teaching and Learning in Information and Computer Sciences **10**(1), 86–90 (2011). https://doi.org/10.11120/ital.2011.10010086
9. Mannila, L., Peltomäki, M., Back, R.J., Salakoski, T.: Why complicate things? introducing programming in high school using python. Conferences in Research and Practice in Information Technology Series **52** (01 2006)
10. Michaelson, G.: Teaching programming with computational and informational thinking. Journal of pedagogic development (Apr 2015)
11. Post, D.E., Goldfarb, O.: Enhancing product innovation with computational engineering. Computing in Science Engineering **19**(6), 4–5 (Oct 2017). https://doi.org/10.1109/MCSE.2017.3971160
12. Szabo, C., Sheard, J., Luxton-Reilly, A., Simon, Becker, B.A., Ott, L.: Fifteen years of introductory programming in schools: A global overview of k-12 initiatives. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research. Koli Calling '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3364510.3364513
13. Wilson, G.: Teaching Tech Together: How to Make Your Lessons Work and Build a Teaching Community around Them. CRC Press (2019), `https://teachtogether.tech/`

A Additional Activities

A.1 Forest fire

This activity proposes simulating a forest fire's dynamics, modeling a forest represented in different stages (tree shoots, lightning strikes, fire spread, cleaning burned trees).

The proposed forest is linear, with N cells or places. There may or may not be a tree in each cell, and if there is a tree, it may or may not be burned. Let's say that two trees are *neighbors* when they are next to each other, in two consecutive cells. The border conditions of our forest are open. Both the first and last cells will have a single neighbor, while all the cells in the middle will always have two neighbors (one to the right and one to the left).

The state of the forest is represented by four stages:

- *Season of sprouts*: Each empty cell has a probability p that a tree will sprout.
- *Season of lightning strike* : with probability f lightning strikes in each cell, and if there was a tree where lightning struck, this tree burns (so the resulting forest will have empty places, live trees, and burned trees).
- *Fire season*: the fire spreads as much as possible. Each burned tree spreads the fire to the immediate living neighboring trees (the one on the right and the one on the left). Note: Propagation ends when there are no burned trees left to spread the fire.
- *Cleaning time*: burned trees are thrown down, and those cells become empty again.

At the end of the four stages, the game begins again by taking the forest's final state as the initial condition for the next first stage.

The questions to be answered in this activity are: What fraction of trees burn for specific values of the parameters p (probability of shoots) and f (probability of lightning strike)? Also, for a fixed probability f of a lightning strike, what is the value of p that maximizes the forest production?

For the hands-on activity, each student receives three cardboard triangles: a red one (to represent a burned tree), a green one (to represent a healthy tree), and a white one (to represent an empty cell). To represent the forest, students and teachers participate by forming a longitudinal forest, in which we all remain connected. Each person will represent a cell in the forest. The forest is open (it does not close on itself). That is, it will have two edges, like the one implemented in the computer.

Initially, each person (representing a cell) generates a random number in `Python`, to decide whether or not a tree will sprout in its place. In each cell, a tree with probability $p = 0.8$ will grow. People representing cells where a tree grew (get a number less than 0.8) hold up the green card. After this step, some people will lift green cards and others lifting white cards (the ones that represent empty cells).

Next, we represent the lightning strike. Those representing cells in which there are trees (those with the green card raised) generate a new random number.

If the number is less than $f = 0.2$, they become burned trees and change the green raised card to a red one (representing the burning tree situation). The trees that did not burn remain with the green cardboard raised. After this instance, some people are lifting green cards, some red, and white (the empty cells).

Next, we represent the spread of fire. If a healthy tree has a burned neighbor, then it burns. On the contrary, if a burned tree has empty cells as neighbors, the propagation in that place stops. At this point, the importance of empty cells that act as firewalls becomes evident. The teacher coordinating the game helps to verify if the propagation is finished or not. After this instance, some people will raise a green card, others a red one, and others a white one (the empty cells).

When summer arrives, the burned trees fall (those with red cardboard take it down) and become empty cells (they lift the white cardboard). When spring arrives, in a new budding season, empty cells can again give rise to trees with probability $p = 0.8$. In those cells where trees should sprout, the student or teacher representing this cell lifts the green cardboard that adds to the previous cycle's green cardboards.

Interestingly, in the hands-on activity, we start from an initial version where each student represents a cell that makes decisions independently (following the rules of the proposed game but without any central organization), to a centralized version coordinated by a teacher who behaves as the orchestrator of all movements. This passage from autonomous systems that perform actions to a programmable centralized system shows that writing the script for this coordinator is enough to represent a complex dynamic.

This activity seeks to strengthen the management of *lists*, access their elements and modify them, based on the value of other elements in the list (the immediate neighbors), generate random numbers to simulate events (like the fall of rays). We also work with the definition of *functions*, implementing conditions and cycles. In this guide for the first time points y vs x are plotted using the `matplotlib` library.

During the implementation, the students have to think about using the computer to represent the following situation: an event occurs with probability p. We discuss the concept of pseudorandom numbers and how to solve the problem of representing an event with probability p using a generated pseudorandom number between 0 and 1 Finally, a major challenge is to develop an algorithm that reproduces the dynamics of the proposed game (in particular, the spread of fire in a longitudinal forest).

A.2 Planetary Orbit

This activity presents a fraction of the solar system, whose simplification consists of two bodies: the Sun, taken as the center of reference, and the earth that orbits around it. The objective of the activity is to be able to calculate this trajectory. We propose the Verlet algorithm to calculate the trajectory of this

orbit [3]. This algorithm uses the current position and the position at a previous time to calculate the next one. The initial data to carry out the simulation are obtained from the NASA website. This model includes different concepts such as discretization of time and decomposition in two coordinates of objects in a plane.

The objective is to calculate the Earth's orbit in such a way as to be able to find 3 points on the said trajectory that determine special days starting from today:

- the next perihelion (the minimum distance between the Earth and the Sun),
- the next aphelion (the maximum distance from the Earth to the Sun),
- each student's next birthday

In this activity, students must use lists where the values corresponding to the two dimensions in which the earth moves (x e y) are stored. The novelty with respect to the previous activities is that the list positions that will be consulted in the next steps have not yet been created. It is important to note that students must understand the change of semantics of each position; a position in one iteration is considered future, in the next one is considered the present one, and in the following, the past.

A.3 Avalanche

This activity consists of modeling a landscape on which snow is falling, and when a certain amount accumulates, it spills over onto some adjacent areas. A square grid (or matrix) represents the terrain. At each time step of the game, one snowflake falls onto the landscape. When a grid's cell reaches 4 snowflakes, it overflows, transferring a snowflake to each of its neighbors (the cells that share an edge in the grid). Therefore, the overflowed cell runs out of snowflakes.

The hands-on activity exemplifies the dynamic, building a human matrix of 4×4. Each student located in the grid identifies who their four neighbors are. The teacher in charge of presenting the activity tosses "snowflakes" one by one (represented by mint candies) at different positions. The teacher intentionally chooses a cell with four neighbors at first. Once the student in that position has obtained four snowflakes (candies in the game), she/he transfers one to each neighbor and is left with none.

Before tossing a new snowflake (candy), all the necessary overflows must be carried out until there is no position in the grid with four candies or more. The game allows the students to note that the grid's bounding cells have fewer neighbors (three or two in the case of corners). During the activity, the teacher clarifies this need for a rule to handle cells with fewer than four neighbors, and the rule established is that each time a cell attempts to transfer a candy to a neighbor cell that does not exist, this candy disappears from the landscape. Thus, the overflowed cell is always emptied.

[3] The Verlet algorithm is a procedure for the numerical integration of second-order ordinary differential equations with known initial values

In this activity, students must use a matrix for the first time, learn to traverse it, and for each position, be able to construct the coordinates that its neighbors represent.

A.4 Predator-Prey Interactions

The predator-prey model represents an interaction in which an individual of one animal species (the predator) hunts an individual of another species (the prey) to survive. In our case, the predators are Lions, and the prey are Antelopes. A square grid represents the landscape. Like the Forest Fire activity (see A.1), the model dynamics include several stages that make up a cycle that is repeated in simulations.

There are three stages that are described below:

- **Feeding**: if a Lion has a neighboring Antelope, it eats it.
- **Reproduction**: if two animals of the same species are neighbors, they reproduce, giving rise to a new animal of that species.
- **Movement**: all animals move.

The same logic of touring the grid step by step, seen in Avalanche, is used to carry out each stages (see A.3). This strategy splits the landscape into two areas at each step: the already addressed area of the current stage and the unaddressed area.

There are cases in which an animal, when displaced, is located in the unaddressed area of the grid, thus being able to participate multiple times in the same cycle or tour of the grid. This anti-intuitive situation serves as a challenge to think about a more complex and more realistic models, at the end of the activity.

The hands-on activity for this class is a small board game (a grid printed on paper) with lion and antelope pieces. Students (in pairs) are told to begin with a specific initial arrangement, carry out a certain number of cycles and later confirm that they arrive at the correct final board arrangement. In this class, students import their completed code into another script (that we provide), which lets them watch their simulated landscapes evolve in real time in a graphical interface. This aspect of the activity invites students to think more like programmers who belong to a community, as they have to make sure their code is implemented exactly as described in the class guide (function names, parameters, output types, etc.) in order for the visualization code to work properly.

A.5 Gases

This activity proposes to model a system of particles contained in a box that interact with each other. For reasons of simplicity and a nicer visualization, the system is limited to two dimensions. This activity introduces the need to write the results to a file in a particular format so that the simulations can be correctly visualized by an external program.

As this is the last activity of the Workshop, the class guide is the least structured, requiring more independent thinking. First, the activity proposes to simulate the movement of a single particle bounded by a box. The trajectory is calculated using a Verlet algorithm, almost the same as that implemented in the planetary orbit practice. The challenge is to keep the particle inside a box by modeling elastic collisions with the walls (students are encouraged to draw this on paper and convince themselves of the correct behavior before implementing their code).

Second, n particles are modeled, all contained in the same box but without interacting with each other. Later, a short-range, repulsive interaction is incorporated between the particles, enabling them to collide and exchange energy. Although the model is relatively simple, it is quite accurate and permits the study of complex processes like phase changes. The diagram in figure 4 summarize the scheme of the code:

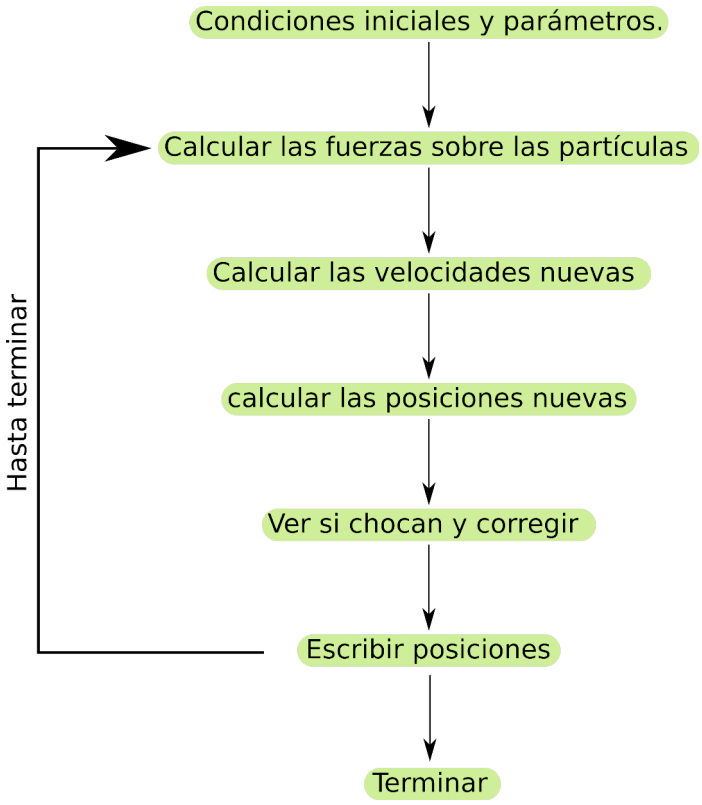

Fig. 4. Scheme of the logic used during the implementation of the model.

The simulation requires multiple nested loops, either through `for` or `while`, which, although students already used them in almost all previous practices, require a higher level of abstraction. For example, each particle's force is calculated by the sum of the interactions with all the others, which requires two nested cycles.

The most significant novelty is the necessity of writing a file with a specific format that a molecular dynamics visualization program called VMD can read: [4].

As a final touch and to verify the generated code, the kinetic energy, and the potential energy are calculated in each simulation step. Students could verify that the total energy, the sum of both, is conserved.

Although the activity is considered complete at this point, there are a variety of extensions that could inspire students to continue playing and learning.

Some extensions are:

- Adding attractive forces
- Implementing a thermostat to keep the temperature fixed.
- Calculate the velocity distribution and check that it corresponds to what Maxwell-Boltzmann predicted (long before the existence of computers).
- Include different types of particles with other interactions depending on the particle type (e.g. positive and negative charges).
- Go from monoatomic to diatomic particles (using a harmonic potential to model the covalent bond).

Like others proposed in this workshop, this activity has different possible culmination stages allowing restless students to develop their capacity and interest by doing more complex optional exercises.

[4] https://www.ks.uiuc.edu/Research/vmd/