Bringing Computational Thinking to non-STEM Undergraduates through an Integrated Notebook Application

Juan Carlos Farah[1], Arielle Moro[2], Kristoffer Bergram[2], Aditya Kumar Purohit[2], Denis Gillet[1], and Adrian Holzer[2]

[1] École Polytechnique Fédérale de Lausanne, Switzerland
{juancarlos.farah,denis.gillet}@epfl.ch
[2] University of Neuchâtel, Switzerland
{arielle.moro,kristoffer.bergram,aditya.purohit,adrian.holzer}@unine.ch

Abstract. Computational thinking courses are no longer exclusive to engineering and computer science students in higher education but have become a requirement in other fields, as well as for students in secondary, primary, and even early childhood education. Computational notebooks, such as Jupyter, are popular solutions to develop the programming skills typically introduced in these courses. However, these solutions often require technical infrastructure and lack support for rich educational experiences that integrate discussion, active feedback, and learning analytics. In this paper, we introduce a web application designed to address these challenges. We present blended learning scenarios supported by this application and evaluate them in an eight-week computational thinking course comprising 67 students pursuing a Bachelor in Business and Economics. We include in our results the impact of the disruption caused by the COVID-19 pandemic, which forced a move from blended to online distance learning for the second half of our evaluation.

Keywords: Computational Thinking · Blended Learning · Digital Education · Jupyter Notebooks · Learning Analytics · Python · COVID-19.

1 Introduction

Computational thinking can be defined as the different thought processes used in computer science to solve problems [35]. Among the main concepts used are modeling problems using abstractions, division of problems into subproblems, design of solutions through sequential steps (algorithms), and identification of patterns. Over the past decade, computational thinking has become a tool to solve problems in virtually every field of study [2]. Learning computational thinking thus becomes crucial not only for engineers and computer scientists, but also for students in domains outside science, technology, engineering, and mathematics (STEM). In line with this trend, computational thinking courses have been introduced not only for a wide array of university degrees, but also in secondary schools [17, 36] and all the way down to early childhood education [3].

An essential part of computational thinking courses is a basic understanding of programming [23]. Due to the complexity of providing a consistent experience across different devices and operating systems, introductory programming courses have traditionally required a technical setup to ensure that all students are running the same development environment [9]. This can be a high barrier to entry for less technical students, teachers, and even institutions lacking proper information technology support. To lower this barrier, computational notebooks have been proposed as a way to minimize the amount of technical setup needed to provide a homogeneous programming environment. Computational notebooks are online tools that combine resources (such as text or images), executable code, and both textual and graphical outputs. They are typically used by data scientists for sharing and keeping track of data exploration as well as for reproducibility purposes [34], and their popularity has "exploded" in recent years, most prominently through the use of Jupyter Notebooks [26].

Nevertheless, self-hosted computational notebook solutions such as Jupyter often require a backend server infrastructure to execute code and manage users, while directing students to cloud-based solutions such as Google's Colaboratory [4] risks violating privacy and legal regulations (e.g., the European General Data Protection Regulation (GDPR) [15]), which many institutions are required to adhere to. With these concerns in mind, we designed a novel web application (app) that allows students to execute Python directly on the browser. This app is free and open source, and can be integrated into online learning platforms — along with collaborative and learning analytics tools—to offer features present in computational notebooks and foster rich learning experiences. To better understand how such an app supports the acquisition of computational thinking skills, we put forth our first research question. **RQ1**: How do non-STEM students in introductory programming courses use and perceive computational notebooks as a tool for learning programming?

To address this question, we incorporated our app into a computational thinking course for students pursuing a bachelor's degree in Business and Economics at the University of Neuchâtel, Switzerland (henceforth the *university*). Our initial goal was to analyze usage and perception of the app within a blended learning scenario. However, at the end of the fourth week of our study, the university had to shut down due to the COVID-19 pandemic, forcing us to adapt the course to a purely online distance learning scenario. This unexpected turn of events prompted a second research question. **RQ2**: How is the usage of computational notebooks different between a blended and a distance learning scenario?

This paper puts forth two main contributions. The first is the design of our app and an overview of how it can be used to create computational notebook learning environments with a lower barrier to entry directed at less technical students and instructors. The second contribution is an analysis of student interaction with the app and the learning environment in which it was deployed (RQ1). This analysis also contains insights on how the switch from a blended learning scenario to a distance learning scenario impacted usage (RQ2).

2 Related Work

One of the key trends in education over the last decade has been the shift to using blended learning models, where traditional face-to-face learning is complemented with digital interaction, whether in-class or at distance [21]. Compared to online learning alone, a blended learning approach has been found to be more effective in terms of learning outcomes [13]. To some extent, most current learning activities occur in some form of blended learning [25, 30]. On top of direct pedagogical gains, blended learning offers the opportunity to integrate learning analytics into the instructor's awareness and reflection processes to assess how students perform and potentially be able to predict student success or failure early on in the course [33].

Introductory programming courses are prime candidates for blended learning [6, 12], and the simplicity and readability of Python have made it an attractive introductory programming language [29]. Although there are a large number of online Python tools available [22], Jupyter Notebooks have become common in introductory Python courses [8, 9, 37]. The combination of an online coding environment that does not require external software and the possibility to run code embedded within text and multimedia content is particularly well-suited to teach computational thinking [24]. Typically among the opportunities offered by tools such as Jupyter is the fact that students can iterate on their coding assignments on the same platform without the need to switch between the assignment and the coding software [28]. Jupyter also includes several tools specifically designed for teaching purposes, such as grading modules [28]. Previous work has explored the usage of online notebooks for teaching computational thinking in different learning activities. For instance, researchers evaluated its usage for (i) lectures, (ii) reading, (iii) homework, and (iv) exams [24].

It should be noted that such notebooks can also have a negative impact on learning, as some argue that they promote poor coding practices because they make it difficult to break code into smaller reusable modules and to develop and run tests on the code [26]. Furthermore, there is a tension between exploration and explanation, as it requires a lot of effort for a user to convert a messy exploratory notebook to a clean shareable notebook [31]. Moreover, such environments still lack support for a wider range of interaction, collaboration, activity awareness, and access control mechanisms [34]. Although computational notebooks are valuable for beginner students, they can be inadequate for experienced users [5, 11]. To address this, notebooks can be personalized according to learning style, programming level, or learning context [24]. Aside from Jupyter, other approaches focus on integrating smart content hosted on different servers to enhance the learning experience [7], while several web-based tools for teaching Python have also been proposed [14, 18, 27].

3 Design

In this section, we present the design of our digital tool and describe how it enables instructors to scaffold computational notebooks and provide a variety of

learning scenarios for their students. We then describe how the tool was used in the context of a course introducing business students to programming.

3.1 Digital Education Tool

As noted in Section 1, we developed an open source web application (henceforth the *code app*[3]) to provide a ready-made Python environment for instructors and students. The code app leverages the Pyodide[4] library to execute Python directly on the browser without any additional dependencies. It supports reading and writing files, receiving input from users, and displaying graphical output from libraries such as Matplotlib [20]. The app also features a command-line interface that serves both to display output and allow students to navigate a virtual file system. In its simplest form, the code app can be used independently of any other software simply by accessing a web link. Nevertheless, it can leverage application programming interfaces (APIs) exposed by digital education platforms to enable advanced features as well as learning analytics. To enable these features and to provide a context resembling computational notebooks, we designed the code app to be compatible with the Graasp open digital education platform [16].

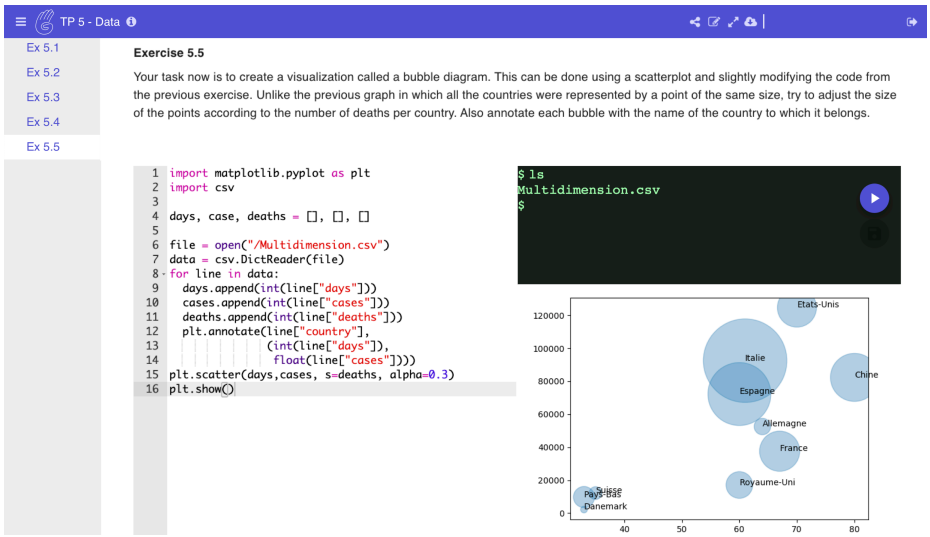

Fig. 1. A computational notebook learning capsule on Graasp. Students can write and execute code, navigate a virtual file system and view graphics.

Graasp provides two interfaces. The first is an *Authoring View* where instructors integrate and configure the resources that they will use to create their online

[3] Code App: github.com/graasp/graasp-app-code
[4] Pyodide: github.com/iodide-project/pyodide

lessons, which we refer to as *learning capsules*. Learning capsules can be scaffolded into step-by-step code exercises, which can be contextualized with text, images, links, chatrooms, and other interactive content. Within this instructor-centric view, the code app can be preconfigured with sample code, data files, and instructions for students. It also features a feedback functionality that allows instructors to review the code of each student and provide comments in a way similar to code reviews on development platforms such as GitHub.

The second is a *Live View*, which is an environment directed at students, accessible through a link. Students can exploit the online lesson, navigating through pages containing the exercises prepared by the instructor. Within this student-centric view, the code app enables students to write, execute and save code, review feedback provided by the instructor, and visualize graphics. The result, as shown in Figure 1, is a computational notebook learning capsule.

During lectures or while watching videos or reviewing slides, students can run code and test results using the code app. The live view also supports a presentation mode, which the instructor can use to guide the students through the learning capsule. Several tools can be included within the learning capsule to provide formative assessment. A simple input app allows students to submit text, while a real-time communication app enables students to spontaneously ask questions and to respond to multiple-choice questions posed by the instructor.

Finally, through the analytics features of the learning capsule, instructors can have an overview of the progress and difficulties students are encountering, and thus adjust their teaching accordingly. As an example, Figure 2 shows a learning dashboard to track user activity. More specifically, it shows the order in which each student has visited the pages available in the live view, as well as the time spent on each of them. If instructors use the live view at the same time, then the instructors' data can be compared against the students' data. Each color represents a page inside the live view. If students were to be perfectly synchronized with the instructor, their color patterns would all be the same.

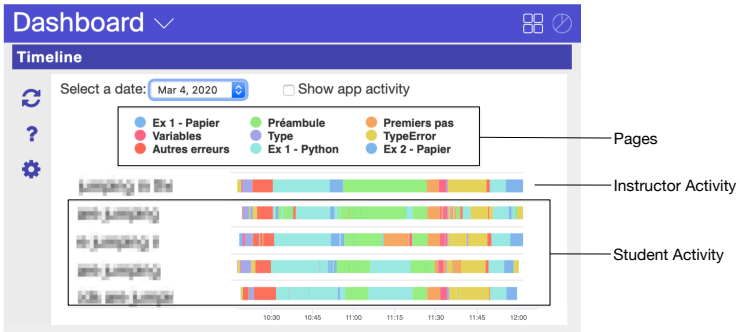

Fig. 2. A learning analytics dashboard to track student use of a learning capsule.

3.2 The Information Technologies Course

The evaluation took place in a first-year course on information technologies for students enrolled in a Bachelor's in Business and Economics. A total of 69 students were enrolled, 31 of them female (45%). Two of the 69 students did not opt in for the study, thus their data was removed. The course lasted one semester (14 weeks) and consisted of two periods (1.5 hours) of weekly lectures and two periods of weekly lab sessions with exercises. Student presence in class was not mandatory. The first half of the semester, which is the focus of this study, covered computational thinking, with two weeks for general theory about concepts (e.g., abstractions, problem division, algorithms) and six weeks of introduction to programming with Python to put the theory into practice (see Table 1). Note that during these eight weeks, teaching was dramatically impacted by the COVID-19 pandemic. Indeed, all in-class lectures were suspended at the end of the fourth week of the semester, and all teaching was moved online. Teaching in the course progressed through three main phases, as outlined below.

Week	Date	Lecture (Wednesdays)	Lab (Thursdays)	Teaching Style	Data Collected
1	17.02	Concepts 1/2	Game	In-Class	Video, Pre-Survey
2	24.02	Concepts 2/2, Python Basics 1/2	Game, Group Activities	In-Class Blended	Video, Activity Traces
3	02.03	Python Basics 2/2	Start Lab 1	In-Class Blended	Video, Activity Traces
4	09.03	Python Lists	Solution Lab 1, Start Lab 2	In-Class Blended	Video, Activity Traces
16.03.2020, COVID-19 Confinement starts. No more in-class lectures or lab sessions after that date.					
5	16.03	Python Functions	Solution Lab 2, Start Lab 3	Distance	Video, Activity Traces
6	23.03	Python Dictionaries	Solution Lab 3, Start Lab 4	Distance	Video, Activity Traces
7	30.04	Python Graphs	Solution Lab 4, Start Lab 5	Distance	Video, Activity Traces
8	06.04	-	Solution Lab 5	Distance	Post-Survey

Table 1. Structure of the Information Technologies Course

Phase 1: Concepts (*in-class*) The first phase covered the first week and a half and consisted mainly of in-class lectures with in-class interactive activities. The exercise sessions were also in-class and focused on getting familiar with algorithmic concepts using a game (Human Resource Machine[5]) as well as through practical group exercises (e.g., designing an analog algorithm to find the most frequent word in a text that was handed out on a piece of paper).

Phase 2: Python (*blended*) The second phase covered the end of the second week and the two weeks that followed, and consisted of blended learning both for lectures and lab sessions. During the lectures, the presentation mode of our learning capsules was used by the instructor. Concretely, the instructor logged in to the live view and moved from one page to the next, typing and executing code while providing explanations. In the meantime, students connected to the same learning capsule but logged in with their own credentials and thus accessed their own version of the exercises, where they could write and execute code

[5] Human Resource Machine: tomorrowcorporation.com/humanresourcemachine

while the instructor was giving the presentation. During the lectures, a real-time communication app was integrated into the learning capsule. Students asked questions and the instructor conducted several polls to see if the level of the course was adequate. During the lab sessions, students were given a learning capsule with five questions, each one containing the code app as well as an input box to provide answers to the questions. The solutions to each exercice were presented the following week. During these first two phases, the lectures were recorded and posted on the university's learning management system (LMS).

Phase 3: Python (*distance*) The third phase was not planned and was triggered by the national response to the COVID-19 pandemic. As the government imposed a partial confinement, the university had to cancel its on-site lectures. For this last phase, lectures were prerecorded and published on the LMS. The instructions for the labs were published on the LMS and a video explaining the solutions was recorded and posted on the LMS a week later. Student interaction with teaching assistants (TAs) and the instructor took place principally (i) through email, (ii) through the communication app, and (iii) through the feedback feature of the code app.

4 Methods

We employed two types of methods to collect data: (i) online activity traces and (ii) surveys. At the beginning of the course, students were informed of our study and were asked to opt in to participate. The learning experience was identical for both those who opted in and those who did not.

We used three forms of activity traces. The first was a measure indicating the number of edits (e.g., keystrokes, deletions, copy/paste actions) that a student performed within the code apps embedded in the learning capsules. We refer to this measure as the *code app interaction* metric. The second form of activity traces was linked to the in-class video recordings and the screencasts for distance learning, which were uploaded to the university's video repository service. The service tracked each time a student logged in and recorded how long a video was played. We refer to this metric as the *time spent watching videos* metric. The third form was generated by tracking how students moved between the different pages of our learning capsules, as visualized in Figure 2.

At the beginning of the course, students were asked to fill in an optional pre-survey about their programming experience and their attitude towards learning technologies. This pre-survey also included an ungraded test assessing their Python knowledge. Students were not informed of their performance. At the end of Phase 3, an optional post-survey was conducted, which included a number of open questions about the learning technologies used, as well as the System Usability Scale (SUS) [1, 19]. The post-survey also included an ungraded post-test. This test followed the same format and included some of the questions posed in the pre-test. This time, students were informed of their performance.

5 Results

In this section, we first present results pertaining to student interaction with the code app, the computational notebook learning capsules, and the accompanying lecture and lab video recordings. We then consider students' perceptions of these learning technologies. Finally, we address the difference between the blended and distance learning scenarios.

How did students interact with the code app? Use of the code app—as measured by the code app interaction metric—varied widely ($\bar{x} = 22167$, $Median = 23990$, $SD = 13223$). However, usage was not significantly correlated with students' self-reported tech-savviness ($r = -0.129$, $p = 0.309$), and a Kruskal-Wallis H test found no significant differences by gender ($H = 0.0228$, $p = 0.880$). The number of days that students interacted with the code app throughout the duration of the course also varied widely across students. On average, students actively interacted with the code app on nine different days ($\bar{x} = 8.925$, $Median = 9$, $SD = 4.831$). As shown in Figure 3, before the switch

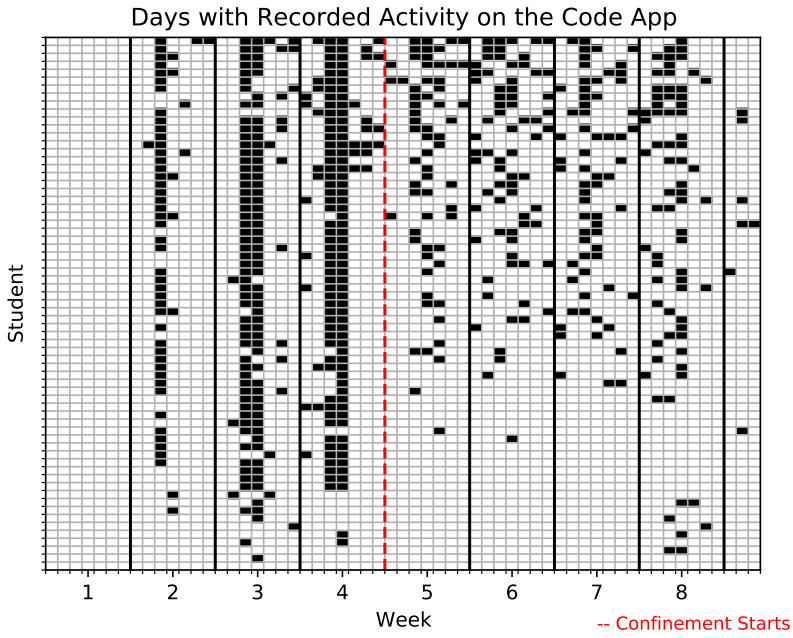

Fig. 3. Days in which there was recorded activity on the code app. Each horizontal line represents a student and each block represents a day in which the student recorded an interaction. A change in use is evident after the switch to distance learning.

from blended to distance learning, usage was concentrated on Wednesdays and Thursdays, coinciding with lectures and labs. Although around a third of students stopped using the code app—or only used it sporadically—after the switch,

around two-thirds of students continued to use it regularly. This pattern is relatively consistent throughout Weeks 5-8 in the same way the long synchronized blocks are present throughout Weeks 2-4. It is worth noting that interaction in the blended scenario was a predictor of interaction in the distance learning scenario ($r = 0.577$, $p < 0.001$).

How usable were the tools? The SUS score ranges from 0 (worst) to 100 (best). The code app achieved a mean score of 71.3 ($n = 57$), which indicates good usability [1]. To assess if students found the other digital tools beneficial for their learning, a three-item questionnaire was disseminated. Questions followed a five-point Likert scale from 'Strongly Disagree' to 'Strongly Agree' ($n = 56$): (i) *I think the use of the interactive slides in Graasp during the course was useful*, (ii) *I think the use of the chat feature in Graasp was useful for the course*, and (iii) *I think the feedback feature in Graasp was useful*. A one-sample Wilcoxon signed-rank test indicated the median for the first item was significantly different from three (the neutral position), $Z = 6.57$, $p < 0.001$, with a very strong effect size ($r = 0.87$). Similar significant results were achieved for the second and third items, with $Z = 4.58$, $p < 0.001$, $r = 0.61$ (strong effect size) and $Z = 4.47$, $p < 0.001$, $r = 0.59$ (moderate effect size), respectively.

We also collected diverse perspectives regarding the tools integrated into the computational notebook learning capsules. Students were presented with an open question: *In your opinion, what are the pros and cons of the digital technologies used in this course?*. A total of 41 students provided open-ended comments. We analyzed responses by articulating emergent themes using line-by-line data coding [10]. Here we discuss the major themes.

1. Easy to use. The first theme that emerged was that the code app was easy to use, easy to understand, and easy to get used to. One student explained: "You get used to the [code app] service quite quickly." Another commented that "the positive points are the ease of use of [the code app], the clarity and the stability of the service".

2. Ready-made. Students also appreciated the ability to execute Python in the code app without any installation requirements. One student commented: "The biggest positive point for me is that we didn't need to install an application. We can easily access the [code app] service. Compared to R, it is easier to use and more modern." Another student noted: "Easy to use / nothing to install / nothing is saved on our computers".

3. Mirroring. A third theme worth noting focused on the possibility for students to mirror what the teacher was doing using the learning capsules. One student reported: "We can put the examples we have seen into practice and we can check by ourselves the explanations given to us during the course work. I find that it puts our computational thinking into practice and motivates us to move forward in this course." Another student commented that "the use of [the code app] during the course allows for a better understanding of the course. You don't just listen, you already assimilate the material".

4. Multiple access. Several students asked for greater flexibility to access multiple learning capsules in parallel. Codes like *multitask* and *parallel control* were recurrent. One student reported: "Can't use multiple learning capsule pages without one page closing." Another student commented that "[the code app] does not open for the lecture and for the lab, you still have to identify yourself and it's painful, it's either one or the other".

To get an overall indication of the qualitative feedback, we performed a sentiment analysis, extracting the polarity (-1 (negative) to 1 (positive)) of each comment. Although results show that the majority of the comments were neutral, this is not surprising, given that we asked students to indicate both positive and negative aspects of the technology used. Nonetheless, our analysis registered a weak positive tendency, indicating that comments were slightly favorable.

How were the computational notebook learning capsules used differently between the blended and distance learning scenarios? Using data from the learning dashboard presented in Figure 2, we examined usage patterns from Week 3 and Week 4 (blended learning), and Week 5 (distance learning) (Figure 4). In the *blended learning scenario*, where the learning platform is used by the instructor and the students at the same time, the dashboard gives the instructor a visual impression of how synchronized students are during lecture. Figure 4 (A) includes all student activity during Week 3, showing that 64 students, including a TA, were active at some point in the lecture. A visual analysis considering only active students—not those who left the class early or arrived late—indicates that there seem to be only 5-6, approximately 10% of all students, who are not following the general page change pattern. Figure 4 (B) shows the dashboard for Week 4. The results are very similar to Week 3, with 61 students active on the platform, most of them—except around 5 students—following the instructor's pattern closely. Note that for both weeks we counted the number of students physically present at the beginning of the second part of the lecture to be 53 (24 female) on Week 3 and 50 (25 female) on Week 4. These student counts are in line with the number of students observed online and convey the fact that the tool was used virtually by all students present in class.

In the *distance learning scenario*, the dashboard allows us to see when students logged in to the online platform to work on the course. Figure 4 (C) shows an overview of the lecture during Week 5 (March 17). The instructor's video recording of the lecture was 35 minutes long (the content was not changed compared to a 90-minute live lecture, however, the online recording did not include interaction time). Once the video was posted, students could access it at their discretion. Figure 4 (C) also shows all activity on the platform related to that particular capsule for the whole week after the video was posted. Overall, 51 students accessed the capsule, and 29 spent at least 30 minutes on the platform. As expected, the usage pattern is not synchronized across students. Nevertheless, there is a diminishing trend of active students per day, with 18 students accessing the learning capsule on the scheduled lecture date (Wednesday, March

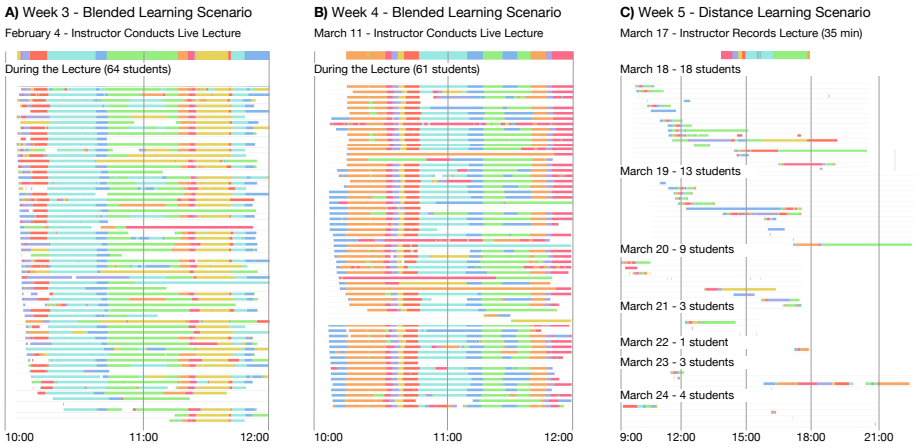

Fig. 4. Comparing blended (A, B) and distance (C) learning scenarios. The instructor's data is show on a thick line above, and each thin line represents a student.

18), 13 students the day after, nine students on the Friday, and a minority of students over the weekend, through to the following Monday.

Given the difference in usage before and after the switch to distance learning, we also wanted to explore the relationship between the code app and other digital tools used for distance education. Specifically, we considered the link between the code app and the video recordings that replaced the live lectures at the start of Week 5. There was a strong positive relationship between the code app interaction and the time spent watching videos metrics, ($r = 0.501, n = 66, p < 0.001$). In other words, the more students used the code app the more time they spent watching videos and vice versa. This might be due to the fact that these resources were meant to be used in parallel to simulate the in-class experience.

6 Conclusion

The results of our evaluation give rise to a number of discussion points concerning our research questions. First, in terms of inclusion, our results show that there is no difference in usage metrics related to either digital literacy or gender. From a learning design perspective, this shows that our code app did not involuntarily discriminate against gender or programming skills. This is particularly important since there can be strong stereotype threats that can hinder learning in computer science, where female students are still widely underrepresented [32]. At the University of Neuchâtel, there are typically around 50% of female students registered in the business curriculum and thus present in all mandatory courses, such as our information technology course. Nonetheless, this percentage drops to under 20% in the elective programming course. It is therefore imperative that the tools employed in introductory courses do not discourage students from continuing studies that reinforce computational thinking skills.

Second, while usage of the code app and the computational learning capsule was synchronized across students and heavily concentrated on lecture and lab days during the blended learning scenario, the distance learning scenario introduced a sharp departure from that pattern of use. Access to the learning capsules became spread over the first two to three days after the lecture was posted, with some students changing their study habits and viewing the material during the evening, as was also highlighted in [12]. Similarly, usage of the code app became scattered throughout the week. Nevertheless, it is important to highlight that around two-thirds of students continued to use the code app regularly throughout the distance learning period. This could signal that the code app and the computational notebook learning capsules could be successfully deployed in both types of learning contexts. Furthermore, usage of the code app during the blended scenario predicted usage in the distance scenario. This could help instructors identify students who might find the switch more challenging and provide them with adequate support, as suggested in [33].

Third, the results from our surveys indicate that students were generally positive regarding the usability and pertinence of both our code app and the computational notebook learning capsules it supports. Moreover, students particularly appreciated the browser-based experience, with no required installation or setup. This serves as a crucial insight into the possible improvements that can be done to the current computational notebook ecosystems in order to lower the barrier to entry for non-STEM students.

To conclude, our study provides a snapshot of how digital tools can allow both instructors and students to adapt to an unforeseen change in pedagogical scenarios. Although our evaluation is by no means conclusive, the fact that usage of our code app was not correlated with prior programming skills and that students found it easy to use is a good sign that it will be positively received in introductory programming courses. In future work, we aim to study how students transition from our code app to more advanced technologies, such as Jupyter Notebooks or integrated development environments. Furthermore, we aim to refine our digital tools following the feedback received and continue to investigate how they can enable computational thinking courses across non-STEM fields.

Acknowledgements

This research has been co-funded by the European Union's Horizon 2020 research and innovation program through the GO-GA Project (grant agreement no. 781012). We would also like to thank Yves Piguet for his contributions.

References

1. Bangor, A., Kortum, P.T., Miller, J.T.: An Empirical Evaluation of the System Usability Scale. Intl. Journal of Human–Computer Interaction **24**(6), 574–594 (2008)
2. Barr, D., Harrison, J., Conery, L.: Computational Thinking: A Digital Age Skill for Everyone. Learning & Leading with Technology **38**(6), 20–23 (2011)

3. Bers, M.U., Flannery, L., Kazakoff, E.R., Sullivan, A.: Computational Thinking and Tinkering: Exploration of an Early Childhood Robotics Curriculum. Computers & Education **72**, 145–157 (2014)
4. Bisong, E.: Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, pp. 59–64. Springer (2019)
5. Borowski, M., Zagermann, J., Klokmose, C.N., Reiterer, H., Rädle, R.: Exploring the Benefits and Barriers of Using Computational Notebooks for Collaborative Programming Assignments. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. pp. 468–474 (2020)
6. Boyle, T., Bradley, C., Chalk, P., Jones, R., Pickard, P.: Using Blended Learning to Improve Student Success Rates in Learning to Program. Journal of Educational Media **28**(2-3), 165–178 (2003)
7. Brusilovsky, P., Malmi, L., Hosseini, R., Guerra, J., Sirkiä, T., Pollari-Malmi, K.: An Integrated Practice System for Learning Programming in Python: Design and Evaluation. Research and Practice in Technology Enhanced Learning **13**(1), 18 (2018)
8. Cardoso, A., Leitão, J., Teixeira, C.: Using the Jupyter Notebook as a Tool to Support the Teaching and Learning Processes in Engineering Courses. In: Intl. Conference on Interactive Collaborative Learning. pp. 227–236. Springer (2018)
9. Chapman, B.E., Irwin, J.: Python as a First Programming Language for Biomedical Scientists. In: Proceedings of the 14th Python in Science Conference (2015)
10. Charmaz, K.: Constructing Grounded Theory: A Practical Guide through Qualitative Analysis. SAGE (2006)
11. Chattopadhyay, S., Prasad, I., Henley, A.Z., Sarma, A., Barik, T.: What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–12 (2020)
12. Chu, Q., Yu, X., Jiang, Y., Wang, H.: Data Analysis of Blended Learning in Python Programming. In: International Conference on Algorithms and Architectures for Parallel Processing. pp. 209–217. Springer (2018)
13. Collopy, R., Arnold, J.M.: To Blend or Not to Blend: Online-Only and Blended Learning Environments. Issues in Teacher Education **18**(2) (2009)
14. 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. pp. 641–646 (2014)
15. European Union: Regulation 2016/679 of the European Parliament and the Council of the European Union. Official Journal of the European Communities **2014**(April), 1–88 (2016)
16. Gillet, D., Vozniuk, A., Rodríguez-Triana, M.J., Holzer, A.: Agile, Versatile, and Comprehensive Social Media Platform for Creating, Sharing, Exploiting, and Archiving Personal Learning Spaces, Artifacts, and Traces. In: The World Engineering Education Forum (2016)
17. Grandell, L., Peltomäki, M., Back, R.J., Salakoski, T.: Why Complicate Things? Introducing Programming in High School Using Python. In: Proceedings of the 8th Australasian Conference on Computing Education - Volume 52. p. 71–80 (2006)
18. Guo, P.J.: Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education. pp. 579–584 (2013)
19. Harrati, N., Bouchrika, I., Tari, A., Ladjailia, A.: Exploring User Satisfaction for E-Learning Systems via Usage-Based Metrics and System Usability Scale Analysis. Computers in Human Behavior **61**, 463–471 (2016)

20. Hunter, J.D.: Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering **9**(3), 90–95 (2007)
21. Johnson, L., Adams Becker, S., Cummins, M., Estrada, V., Freeman, A., Ludgate, H.: NMC Horizon Report: 2013 Higher Education Edition (2013)
22. Kim, A.S., Ko, A.J.: A Pedagogical Analysis of Online Coding Tutorials. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. pp. 321–326 (2017)
23. Lye, S.Y., Koh, J.H.L.: Review on Teaching and Learning of Computational Thinking Through Programming: What is Next for K-12? Computers in Human Behavior **41**, 51–61 (2014)
24. O'Hara, K., Blank, D., Marshall, J.: Computational Notebooks for AI Education. In: The Twenty-Eighth International Flairs Conference (2015)
25. Oliver, M., Trigwell, K.: Can 'Blended Learning' Be Redeemed? E-learning and Digital Media **2**(1), 17–26 (2005)
26. Perkel, J.M.: Why Jupyter is Data Scientists' Computational Notebook of Choice. Nature **563**(7732), 145–147 (2018)
27. Pritchard, D., Vasiga, T.: CS Circles: An In-Browser Python Course for Beginners. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education. pp. 591–596 (2013)
28. Project Jupyter, Blank, D., Bourgin, D., Brown, A., Bussonnier, M., Frederic, J., Granger, B., Griffiths, T., Hamrick, J., Kelley, K., Pacer, M., Page, L., Pérez, F., Ragan-Kelley, B., Suchow, J., Willing, C.: nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook. Journal of Open Source Education **2**(16), 32 (2019)
29. Radenski, A.: "Python First": A Lab-Based Digital Introduction to Computer Science. SIGCSE Bulletin **38**(3), 197–201 (2006)
30. Rodríguez-Triana, M.J., Prieto, L.P., Vozniuk, A., Boroujeni, M.S., Schwendimann, B.A., Holzer, A., Gillet, D.: Monitoring, Awareness and Reflection in Blended Technology Enhanced Learning: A Systematic Review. International Journal of Technology Enhanced Learning **9**(2-3), 126–150 (2017)
31. Rule, A., Tabard, A., Hollan, J.D.: Exploration and Explanation in Computational Notebooks. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. pp. 1–12 (2018)
32. Starr, C.R.: "I'm Not a Science Nerd!": STEM Stereotypes, Identity, and Motivation Among Undergraduate Women. Psychology of Women Quarterly **42**(4), 489–503 (2018)
33. Van Goidsenhoven, S., Bogdanova, D., Deeva, G., vanden Broucke, S., De Weerdt, J., Snoeck, M.: Predicting Student Success in a Blended Learning Environment. In: Proceedings of the 10th International Conference on Learning Analytics & Knowledge. pp. 17–25 (2020)
34. Wang, A.Y., Mittal, A., Brooks, C., Oney, S.: How Data Scientists Use Computational Notebooks for Real-Time Collaboration. Proceedings of the ACM on Human-Computer Interaction **3**(CSCW), 1–30 (2019)
35. Yadav, A., Stephenson, C., Hong, H.: Computational Thinking for Teacher Education. Communications of the ACM **60**(4) (2017)
36. Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., Korb, J.T.: Introducing Computational Thinking in Education Courses. In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. pp. 465–470. ACM (2011)
37. Zastre, M.: Jupyter Notebook in CS1: An Experience Report. In: Proceedings of the Western Canadian Conference on Computing Education. pp. 1–6 (2019)