MOON: Assisting Students in Completing Educational Notebook Scenarios

Christophe Casseau, Jean-Rémy Falleri, Thomas Degueule, Xavier Blanc

▶ **To cite this version:**

Christophe Casseau, Jean-Rémy Falleri, Thomas Degueule, Xavier Blanc. MOON: Assisting Students in Completing Educational Notebook Scenarios. 2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Oct 2023, Washington DC, United States. pp.157-167, 10.1109/VL-HCC57772.2023.00026 . hal-04219684

HAL Id: hal-04219684
https://hal.science/hal-04219684

Submitted on 27 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MOON: Assisting Students in Completing Educational Notebook Scenarios

Christophe Casseau*, Jean-Rémy Falleri*†, Thomas Degueule* and Xavier Blanc*
*Univ. Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR5800
Talence, France
name.surname@labri.fr
†Institut Universitaire de France

***Abstract*—Jupyter notebooks are increasingly being adopted by teachers to deliver interactive practical sessions to their students. Notebooks come with many attractive features, such as the ability to combine textual explanations, multimedia content, and executable code alongside a flexible execution model which encourages experimentation and exploration.**

However, this execution model can quickly become an issue when students do not follow the intended execution order of the teacher, leading to errors or misleading results that hinder their learning. To counter this adverse effect, teachers usually write detailed instructions about how students are expected to use the notebooks. Yet, the use of digital media is known to decrease reading efficiency and compliance with written instructions, resulting in frequent notebook misuse and students getting lost during practical sessions.

In this article, we present a novel approach, MOON, designed to remedy this problem. The central idea is to provide teachers with a language that enables them to formalize the expected usage of their notebooks in the form of a script and to interpret this script to guide students with visual indications in real time while they interact with the notebooks.

We evaluate our approach using a randomized controlled experiment involving 21 students, which shows that MOON helps students comply better with the intended scenario without hindering their ability to progress. Our follow-up user study shows that about 75% of the surveyed students perceived MOON as rather useful or very useful.

***Index Terms*—educational notebooks, notebook scenarios, Jupyter notebooks**

I. Introduction

Teachers are increasingly using Jupyter notebooks as a support for educational activities such as graded assignments [1, 2, 3], interactive textbooks [4], interactive exercise worksheets, or live coding [5]. The primary advantage of a notebook lies in its ability to gather all the essential tools within a single platform. This allows teachers to provide students with an interactive document that includes text and instructions, as well as executable code, images, and videos. The advantages of using notebooks in an educational context are assessed not only in scientific disciplines such as physics [6, 7], chemistry [8, 9, 10, 11, 12], mathematics [13] or computational sciences [14, 15, 16, 17], but also in other disciplines such as humanities for example [18].

978-1-6654-4592-4/21/$31.00 ©2023 European Union

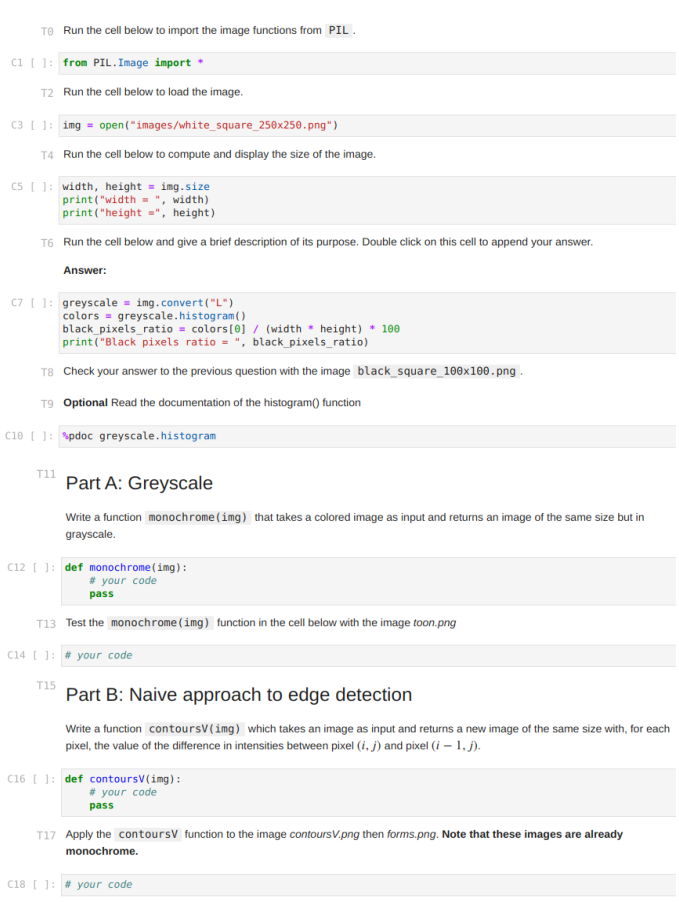
Writing instructions for a notebook-based learning activity is challenging. It is notoriously difficult to write instructions for digital media as studies have shown that reading efficiency in such media is lower than for paper-based support [19, 20, 21, 22]. Besides, notebooks are composed of cells (text or code) that can be read and executed in any order, which can lead to non-deterministic notebook behavior. This non-determinism, which has been widely studied in the context of data science [23, 24, 25, 26], can be very problematic in an educational context and must be carefully taken into account by the teacher instructions.

Notebooks offer a unique opportunity to provide more integrated instructions since all the material required for the learning activity is at the fingertip of the students on the same platform. Building on this idea, we designed MOON, an approach intended to operationalize and enhance the efficacy of the instructions written by a teacher in a notebook-based learning activity. This approach is enabled by two main contributions. First, we provide a DFA-based language that allows teachers to express in a script how students are supposed to manipulate the different cells of a notebook. Second, we materialize the script with a color system complemented by emojis that assists students directly inside notebooks (Figure 1b). The purpose of this assistance is to enable students to maintain a consistent notebook without preventing them from engaging in exploratory activities.

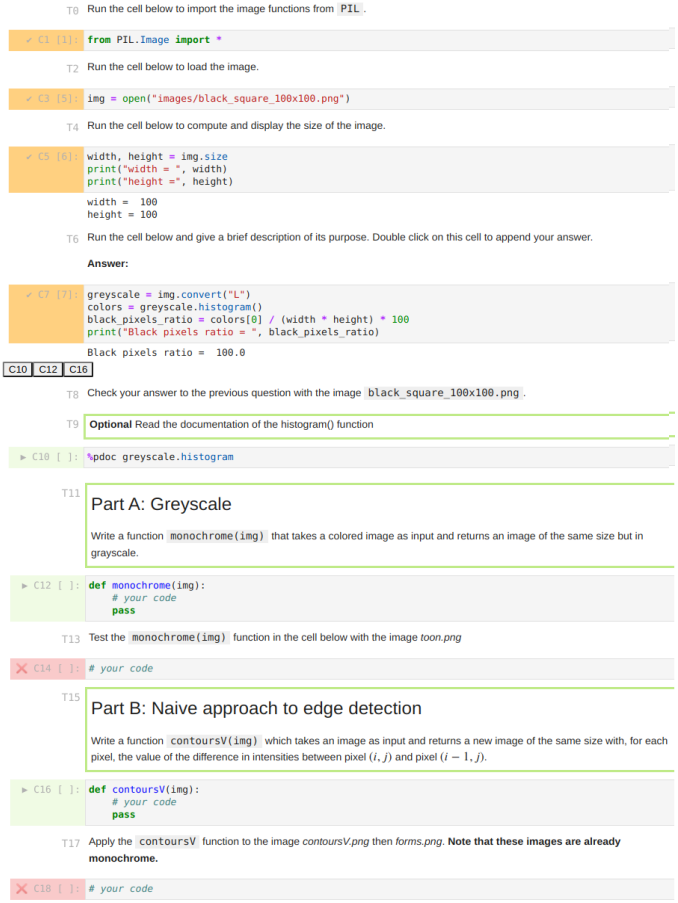
To evaluate the usefulness of our approach, we conducted a randomized controlled study involving first-year computer science students at the university in a three hours programming practical session on graph theory. We observe that students in the control group (without MOON) complete as many cells as those using MOON. However, students in the control group have much greater difficulty following the written instructions provided by the teacher. We also proceeded to a user study that confirms the findings of our controlled experiment and shows that most students found MOON useful. The implementation of MOON, the notebooks used in the evaluation, and the raw data of the controlled study and user study are available in the accompanying Zenodo artifact [27].

II. Educational Notebooks

There are multiple ways to use notebooks for teaching. Our study primarily focuses on interactive educational notebooks



(a) Example of an educational notebook embodying the teacher's scenario.



(b) State of the notebook after the student is run in the following order the cells : `C1 C3 C5 C7 C3 C5 C7`

Figure 1: All cells of the notebook are labelled `Ci` for code cells and `Tj` for text cells, where `i` and `j` represent the indices of the notebook cells.

used as course and exercise sheets, exams and assignments. To create them, the teacher must write a notebook containing instruction cells related to the learning activity, as well as the instructions that will guide students in the notebook's execution model. The written content constitutes the notebook's scenario. Consider as an example an introductory image processing course in Python where students are asked to experiment with `Pillow`, an imaging library, and to implement well-known algorithms. The teacher decides to organize the session around three parts. Note that we present a simplified scenario compared to the one given to the students. In the first part, students should import the library, load a sample image, compute its size and black pixel ratio, and try again with another image. In the second part, students should implement a simple algorithm to convert an image to grayscale and showcase its use. In the third part, students should implement a naive algorithm for edge detection and showcase its use.

Once the learning activity is finalized, the teacher materializes it in the notebook intended to be used by students during the session. The challenges when writing the notebook's scenario are twofold: i) crafting text that effectively imparts knowledge and ii) providing clear directions for students to execute the code cells in the intended sequence. Indeed, since notebook cells may be executed in any order, students may encounter issues if they attempt to use variables and functions before they are defined, for instance. The implications for learning are significant, as students may confront execution errors and exceptions that divert their attention—and the teacher's!—from the pedagogical objectives. Once the notebook is completed, the text and code cells should make it straightforward for students to follow the scenario written by the teacher.

Figure 1a depicts an educational notebook with a scenario. For the sake of readability, each cell is associated with a unique identifier corresponding to its position in the notebook. The letter `C` denotes code cells while the letter `T` denotes text cells. As the instructions in the first part suggest, the students are asked to execute cells `C1`, `C3`, `C5`, and `C7` sequentially and then optionally execute cell `C10`. After executing `C7`, the students should edit `C3` to load another image and re-run the intermediate

cells C5 to check their answers. The remainder of the notebook is divided into two parts which can be worked on independently.

When students open the educational notebook, they must read the text cells to follow the scenario. Many studies have delved deeply into reading comprehension models and have shown that comprehension depends on many factors, such as reader characteristics, text content and design, and reading instructions [28]. A recent study even clearly demonstrates that reading comprehension scores are lower for digital texts than for print texts, regardless of age [22]. Even with the best intentions, students may inadvertently end up with a notebook state not intended by the teacher. Imagine a notebook in which the student executes cells C1, C3, C5, and C7 sequentially and then changes the image used in cell C3, as requested. However, they do not re-execute cell C5, thinking it is only intended to display the image's dimensions, and instead execute C7 directly to compute the black pixel ratio. The obtained result is incorrect because the black pixel ratio for the *black_square_100x100.png* image is calculated with the height and width dimensions of the *white_square_255x255* image that has not been updated (cell C5). Without immediate feedback on the issue, there is a high risk that the student will not notice the issue and proceed, which may trigger other problems down the line and hurt their learning. Finally, because notebooks foster live and exploratory programming, students may insert new code cells anywhere in the notebook to try things out, delete cells, or modify existing cells without ensuring the consistency of the entire notebook. It is thus easy for students to corrupt their notebooks, sometimes without noticing, which hurts the consistency of the results and the intended scenario.

To address these issues and support students in manipulating educational notebooks, this paper introduces a novel approach, MOON, fully integrated with Jupyter, that provides the teacher with a simple language for scripting their scenario within the notebook. The resulting scripts are materialized while students interact with their notebooks, using visual assistance to guide them at every step by identifying the following cells to execute.

III. MOON Design

Our approach, MOON, has two main objectives: i) to allow the teacher to express the scenario they have designed for their notebook in the form of a script and ii) to interpret the script to enrich the notebook with visual indications for the student identifying the next cells to be executed and information on past or upcoming executions of all the cells.

A. From Scenarios to Scripts

A scenario is embodied by the code and text cells of the educational notebook. To better understand the execution patterns of code cells used by teachers in practice, we conducted an informal experiment. We analyzed about one hundred Jupyter notebooks on GitHub and Kaggle that included instructions for the reader. Our corpus of notebooks primarily consisted of educational notebooks, mostly from university courses. These notebooks were selected to cover a wide range of academic disciplines and subjects, to try and have comprehensive collection of educational materials (exercices worksheets, assignments and interactive textbooks). We identified three main patterns for cell execution: i) linear execution, ii) non-linear execution, and iii) optional execution. We then created a simple scripting language that covers these three execution models and allows the teacher to define the intended sequence of execution of code cells in their notebook. We limit ourselves to these three patterns as they are sufficient to express all the scenarios we encountered.

The basic construct manipulated in our scripting language is the code cell, each optionally associated with a set of text cells containing the corresponding instructions. Code cells are written as follows: Ci∼Tj∼...∼Tn, where $i, j, n \in \mathbb{N}$ denote the indices of the cells involved in the notebook. For example, in the notebook of Figure 1a, C1∼T0 denotes the code cell C1 and its associated instructions in text cell T0. In the rest of the article, we omit textual cells from the scripts for the sake of clarity. The execution order for the entire notebook is then specified by combining cells using three operators derived from the execution patterns above:

a) The linear pattern: The linear pattern is naturally the most common, as it matches the standard top-down reading order. To express that a set of code cells should be executed linearly, our script language uses the parentheses operator (). The first part of the teacher's scenario involves a linear execution pattern that requires the following cells to be executed sequentially: C1 C3 C5 C7 C3 C5 C7. Using the scripting language, this is expressed as (C1 C3 C5 C7 C3 C5 C7).

b) The non-linear pattern: The non-linear pattern indicates that a set of cells may be executed in any order, using the square brackets operator []. For example, with two code cells noted Ci and Cj we write [Ci Cj] which gives two possibilities of execution Ci then Cj or Cj then Ci.

c) The optional pattern: The optional pattern, denoted with a question mark, indicates that a (set of) code cells may or may not be executed. In our example, the first part of the notebook is expressed as C1 C3 C5 C7 ?C10, with C10 optional.

Most importantly, the scripting language makes it possible to compose these patterns. To illustrate this combination of operators, consider the notebook of Figure 1a in its totality. The student must start with the first part, and then the remaining two (Part A and Part B) may be done in any order. The first part is written as (C1 C3 C5 C7 C3 C5 C7 ?C10) in the script with C10 an optional cell. The script for parts A and B are as follows: A→(C12 C14) and B→(C16 C18). The three parts are put together by combining their scripts with the linear and any order patterns, as follows: ((C1 C3 C5 C7 C3 C5 C7 ?C10)[(C12 C14)(C16 C18)]). Parenthesis or bracket operators can be prefixed with a question mark to indicate optional cells.

B. Assisting Students in their Manipulation of the Notebook

Once the teacher has written the script implementing their scenario, it is fed into MOON to guide students while they are working on their notebooks. MOON uses a three-color coding

system to guide students in realizing the scenario. Specifically, at every step, MOON highlights the cells that can be executed in green, cells that have been completed in orange and cells that are not yet ready for execution in red. In the example depicted in Figure 1b, we can see that cells `C1`, `C3`, `C5` and `C7` are colored in orange, indicating that the student has already executed them. MOON indicates that three cells, highlighted in green, can now be executed: the optional cell `C10`∼`T9`; Part A with the code cell `C12`∼`T11`; and Part B with the code cell `C16`∼`T15`. Note that associated text cells are also highlighted to guide the student to the cells that contain the instructions for the different tasks that are immediately accessible. If the student ignores the optional code cell `C10`, it turns red at the next step, whereas if they execute it, it turns orange. Tasks that are not yet accessible and should not be executed are colored in red. If a student executes a red cell, the colors of the cells remain unchanged. The only way to progress in the scenario and highlight the next set of green cells is to execute one of the green cells that denote a task. These visual indicators inform the students about their progression in the notebook.

Additionally, MOON decorates the last executed cell with buttons pointing to the next possible cells to execute. This feature serves two primary objectives: i) to give an overview of the next possible cells without having to scroll through the page if they are outside the visible screen area; ii) to allow the students to quickly navigate through the notebook by clicking on the button corresponding to the desired code cell. In our example, there are three buttons to indicate and jump to the three cells that can now be executed, corresponding to the optional cell (`C10`), Part A (`C12`), and Part B (`C16`).

C. Support for Exploratory Programming

Jupyter notebooks are a powerful tool for education due to their support for exploratory programming, enabling students to learn and experiment with code with great flexibility. At first glance, MOON considerably reduces this flexibility by forcing students to follow the teacher's scenario. However, to still encourage exploration, MOON accounts for the cases where students re-execute past cells and insert new cells while keeping the intended scenario on track.

1) Automatic Backtracking: In the exploratory phase, students may modify and re-execute the cells they have already executed, typically to correct their code or explore alternatives. In MOON, this means that a student may execute an orange cell which is not one of the cells planned in the scenario at this point. These exploratory phases are necessary for the students but potentially dangerous for the execution model of the notebook, as shown in the example of Figure 1b. In this case, MOON offers an *automatic backtracking* menu that assists the student in resuming the teacher's scenario as seamlessly as possible. When executing an orange cell, MOON puts the students back to the last execution of that cell intended by the script and updates the colors of all cells in the notebook accordingly. While the colors assigned to the different cells will be updated, the memory state of the notebook will not, and the students must manage the potential issues.

Looking back at our example with the following script `(C1 C3 C5 C7 ?C10)`, when the student decides to use the image of the black square to check the ratio of black pixels, they forget to validate cell `C5`, yielding surprising results for cell `C7`. With MOON, the execution of the orange code cell `C3` puts the student back in the script by coloring cell `C5` in green again. The execution of cell `C5` allows obtaining the update of the variables `width` and `height` of the image *black_square.png* and the green highlighting of code cell `C7`. When the student executes cell `C7`, they now obtain the correct result for the ratio of black pixels. Finally, MOON also includes a *back* button that allows students to go back one step in the sequence of the script.

2) Adding and Deleting Cells: MOON retains the ability to add new code and text cells to support active reading. For example, students can create additional code cells beyond those provided by the teacher, which can serve as exploratory cells, or create text cells that can be used for note-taking, allowing them to keep all their work in a single document. When one of these cells is created, it remains white to distinguish it from the cells in the scenario. MOON also adjusts the indices of cells belonging to the scenario in the script, if necessary. However, none of the cells involved in the scenario can be deleted once the script has been loaded.

IV. Implementation

MOON is implemented as a JupyterLab plugin that takes as input a script as defined in the previous section. Note that our scripts are regular expressions, except for non-linear patterns. We convert these patterns to obtain all the possible combinations of corresponding linear patterns and produces a deterministic finite automaton (DFA) using the algorithm described in [29]. Therefore, it is not possible to have a scenario consisting of too many cells that could be executed in any order due to exponential complexity. To construct the AST of a script, we use tsPEG[1], a PEG Parser generator designed for TypeScript. In the context of a notebook, the DFA is a model that represents a set of states and transitions between them, based on code cell executions. If we take our notebook as an example (Figure 1a) and the following script: `C7 ?C10 [(C12 C14) (C16 C18)]`, we build the DFA shown in Figure 2.

In addition to the DFA, we also maintain a user trace which is a sequence of pairs $(\texttt{Ci}, q_j)$ that corresponds to the list of valid transitions `Ci` leading to a state q_j done by a given user. Note that invalid transitions are not recorded in this trace. For instance, if a user executes `C7`, `C12` and `C18` for the previously described notebook and script and starting at q_0, the user trace will contain: $(\texttt{C7}, q_1), (\texttt{C12}, q_3)$.

When the user executes a code cell, the automaton checks if this execution corresponds to a possible transition for the current state. If the transition is allowed, the automaton changes state and the colors of the notebook cells (code and text) are updated. For our example (Figure 2), consider the automaton in state $q1$ with a user trace containing $(\texttt{C7}, q_1)$.

[1]https://github.com/EoinDavey/tsPEG

All cells belonging to the user trace are orange. There are three transitions allowed by the automaton. The execution of the code cell `C10` which is optional and the execution of the code cells `C12` and `C16` in any order. These three code cells, which represent the next possible tasks, are green, as are their associated text cells and the other code cells are red (Figure 1b). Let's now look at what happens depending on whether or not the optional code cell `C10` is executed. If the user validates the transition associated with the cell `C10`, it is added to the trace $(\texttt{C7}, q_1), (\texttt{C10}, q_2)$ and turns orange. In this case, there are now two possible transitions: the execution of code cells `C12` and `C16`, whose cells are colored green on the notebook. The other code cells are red. If the user validates the transition associated with the code cell `C12` without having validated the transition associated with the code cell `C10`, the user trace becomes $(\texttt{C7}, q_1), (\texttt{C12}, q_3)$ and the code cell `C12` becomes orange. In this case, the next authorized transition is associated with code cell `C14` (green) and the code cells `C10` and `C16` turn red on the notebook.

We also implemented the possibility for the student to re-run an orange cell and place the automaton in the state that corresponds to this transition in the script. For example, if we have the following user trace: $(\texttt{C7}, q_1), (\texttt{C12}, q_3), (\texttt{C14}, q_4)$, on the notebook the next task is designated by the green code cell `C16`. Imagine now, that the student changes the orange code cell `C12` and then executes it. The automatic backtracking looks in the user trace for the last pair containing the transition associated with the `C12` code cell by deleting all the next ones in the trace. Finally the user trace contains only $(\texttt{C7}, q_1), (\texttt{C12}, q_3)$ and the DFA is resetted to state q_3. The code cells `C7` and `C12` are orange, the code cell `C14` is green and other code cells are red.

Let's illustrate another possible situation with the following new user trace: $(\texttt{C0}, q_1), (\texttt{C2}, q_2), (\texttt{C0}, q_3), (\texttt{C4}, q_4)$, with a DFA in state q_4. From this trace, we can observe that the scenario requires the student to execute code cell `C0` twice, first at the beginning and then after executing code cell `C2`. If the student decides to execute code cell `C0` again, the backtracking implementation will reset the DFA to state q_3 and code cell `C4` will turn green. However, with backtracking alone, it is not possible to reset the DFA to state q_1. To overcome this limitation, we have also implemented a "back" button that allows users to reset the DFA to the previous state in the user trace. Each click on the back button removes the last pair in the user trace, resets the DFA to the previous state, and updates the cell colors in the notebook. Once the DFA is in state q_4, it is sufficient to click the back button four times to reset the DFA to state q_1. In a more complex user trace, it is also possible to combine automatic backtracking with the back button to return to a previous state.

V. Controlled Experiment

To evaluate the ability of MOON to support students in manipulating educational notebooks with a scenario, we conducted a controlled experiment. Specifically, our evaluation aims to answer the two following research questions:

- RQ1 Does MOON help the students to better comply with an educational notebook with a scenario?
- RQ2 Does MOON hinder the ability of students to progress in the scenario?

Our hypothesis for RQ1 is that the assistance provided by MOON might decrease the amount of incorrect usage of the notebook. Our hypothesis for RQ2 is that the assistance provided by MOON might make it longer to manipulate the notebook and hinder the students' progression.

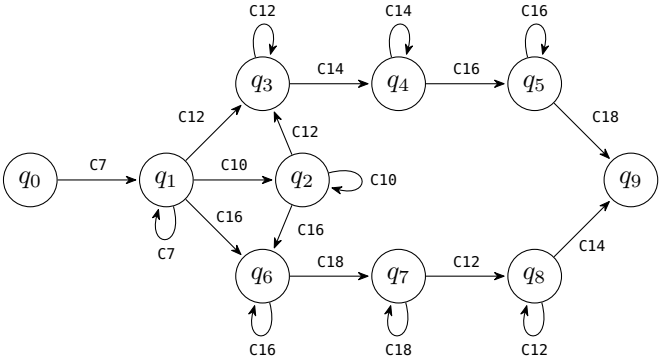


Figure 2: Automaton obtained from the script: `(C7 ?C10 [(C12 C14)(C16 C18)])`

A. Experimental Design

To study the impact of MOON, we proceeded to an A/B testing controlled experiment, where one group of students is given a notebook without MOON, and the other group is assigned the same notebook with MOON support.

We conducted our experiment as part of an introductory programming course involving 21 first-year bachelor students from the university of Bordeaux during the first semester. This course is organized around a set of chapters (*e.g.*, arrays, sorting algorithms, images, graph theory). Each chapter is implemented as a Jupyter notebook file which interweaves course concepts, images, source code, and coding questions. The study took place in the first week of January 2023, the students had already manipulated notebooks during 6 x 2H40 practical sessions, and the study took place in the 7th (last) session. Hence, the students already had experience with notebooks.

For the experiment, we randomly assigned 11 students to the control group and 10 students to the experimental group that uses MOON. The students using MOON received a short twenty-minute training session where one of the authors, who was the instructor of the students, explained the meaning of the colors (Section III-B) and the features of the backtracking menu (Section III-C) on a sample notebook.

The notebook used in this experiment consists of 3 exercises on graph theory with different cell execution patterns [27]. The first exercise follows a linear execution pattern with an optional cell. The second exercise combines linear and non-linear execution patterns. The third exercise only involves a

linear execution pattern. The last two exercises can be done independently, so the student may start with either one. The complete notebook requires 23 cell executions distributed across 20 code cells and 23 markdown cells: three code cells are expected to be executed at least twice. Each code cell requires the student to either execute, complete, or fill in the cell. This notebook was distributed to the students during a 2h40 practical session. The session's instructor was tasked to only answer questions related to the course's content (graph theory) and not to assist students in manipulating the notebook to avoid biasing their behavior. At the end of the practical session, the instructor gathered the notebooks produced by the students.

To answer our research questions, we monitored how the students manipulate the notebook. To that extent, we instrumented JupyterLab for both groups to record a complete trace of the students' cell execution actions on the notebook. This trace, the log trace, is saved into the metadata of the notebooks. Note that the log trace differs from the user trace described in Section IV because i) it records every cell execution, even those not complying with the scenario, and ii) it is never modified, even in case of backtracking. Finally, we simplify the log traces by removing and replacing sequences of execution of the same cell with a single execution of this cell. We perform this simplification because it is common for students to execute multiple times the same code cell when working on it. Still, it does not affect compliance with the intended scenario.

To study RQ1, we introduce a metric to capture the number of incorrect usage of the notebook. Our metric is therefore defined based on the concept of *deviation* from the scenario, where a deviation is the execution of a red cell, *i.e.*, the execution of a cell that is not permitted in the scenario. To compute whether the actions recorded in the log traces correspond to a valid (green) transition, a backtracking (orange), or an invalid transition (red), we replay the students' log traces offline with MOON after the practical session. Finally, to define our metric, called *fitness*, we note g, o, r the total number of green, orange, and red cells executed in a student trace. Fitness is defined as the ratio of green or orange cell executions (*i.e.*, valid executions) over the total number of cell executions: $\text{fitness} = (g + o)/(g + o + r)$

A fitness of 1 indicates that a student has never executed a red cell, while a value of 0 indicates that a student only executed red cells. To assess the differences between the control and experimental groups, we use a two-sided non-parametric Mann-Whitney U test with a p-value cutoff of 0.01 and report the Rank Biserial Correlation as an effect size. Our null hypothesis is that the fitness is the same in both groups, and our alternate hypothesis is that the fitness is different in both groups.

To study RQ2, we introduce a metric to capture how far the students went in the notebook. To that extent, we define *completeness* as the number of distinct cells executed in a given log trace. A student that did not execute any cell of the notebook corresponds to a completeness of 0, while a student that executed every possible cell inside the notebook has a completeness of 20. To assess the difference between the control and experimental groups, we use a two-sided non-parametric Mann-Whitney U test with a p-value cutoff of 0.01, and we report the Rank Biserial Correlation as an effect size. We also evaluate the students' code by assigning a grade to each notebook. Our null hypothesis is that the completeness and/or correctness is the same in both groups, and our alternate hypothesis is that the completeness and/or correctness is different in both groups.

B. Results

1) RQ1: Figure 3a depicts the fitness of the log traces from both groups. Visually, we note a sharp difference between the two groups. The fitness in the control group varies between about 0.25 (meaning only a quarter of the executions are in a green or orange cell) to about 0.75. In the experimental group, the fitness values are very close to 1 for all traces, with three traces having a fitness of 1 (no execution of a red cell). The Mann-Whitney U test yields a p-value $p = 0.000122$ and an effect size $rbc = -1$. We thus reject the null hypothesis and accept the alternate hypothesis. The effect size indicates a clear effect in favor of the group using MOON, with no fitness value from the control group (maximum value 0.8) exceeding the lowest fitness value in the experimental group (0.91). In conclusion, the data support the hypothesis that MOON reduces the amount of incorrect notebook usage.

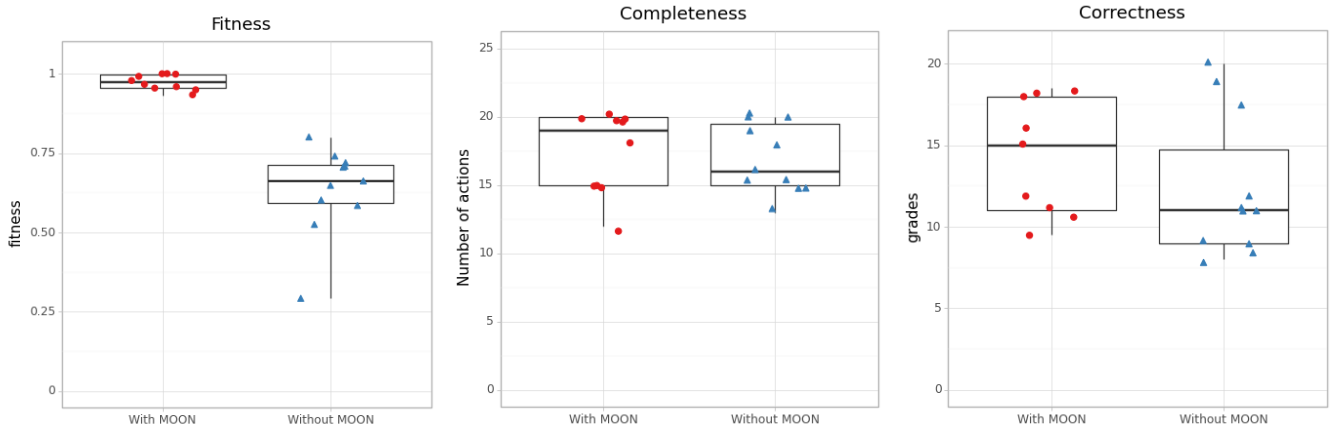
2) RQ2: Figure 3b depicts the completeness computed from the log traces from both groups and figure 3c shows the students' scores for the practical session. For this two figures, visually, the distribution of the values is similar between the two groups, with a comparable spread and a slightly lower median in the control group. For completeness, the Mann-Whitney U test yields a p-value $p = 0.53$ and an effect size $rbc = -0.145455$. Thus, we cannot reject the null hypothesis that the completeness is the same between the groups. For Correctness, the Mann-Whitney U test yields a p-value $p = 0.29$ and an effect size $rbc = -0.292929$. Thus, we cannot reject the null hypothesis that the grades are the same between the groups. The calculation of the median reveals a 4-point difference in favor of the group with MOON. Similarly, when calculating the average, the experimental group has a mean of 14.3 with a standard deviation of 3.6, while the control group has a mean of 12.4 with a standard deviation of 4.4. Moreover, in both cases the effect size is small and in favor of the experimental group. Therefore, the data does not support the hypothesis that MOON hinders students' progression. On the contrary, there is a slight advantage in the experimental group.

In summary, based on our controlled experiment, we answer our research questions as follows:

> MOON helps students to better comply with the notebook's scenario without hindering their ability to progress in the scenario.

C. Threats to Validity

Regarding construct validity, the completeness metric might not accurately capture the actual progress of students inside the



(a) Fitness in the control and experimental groups

(b) Completeness in the control and experimental groups

(c) Correctness in the control and experimental groups (french grading system [0-20])

Figure 3: Analysis of student notebooks for experimental (with MOON) and control groups (without MOON).

notebook, as executing a cell is different from completing the work required in this cell. To counter this threat, the instructor reviewed all notebooks and ensured there were no discrepancies between the log trace and the completed cells in the 21 notebooks. We examined the correctness of the students' notebooks, but we had reservations about the subjectivity of grading since it was done by one of the authors.

Regarding internal validity, JupyterLab's ability to restart the execution kernel can severely disrupt the use of MOON since it resets the notebook's memory while keeping the state of MOON. Therefore, the instructor asked the students to reset MOON if they had to restart the kernel, but some students could have ignored this instruction. The instructor also advised students to only click on the refresh button on the browser if they were sure they had saved their notebooks. This action resets the plugin and clears the user trace to start a new one. MOON makes heavy use of colors to communicate feedback to students, which might be an issue for some students due to *e.g.*, color blindness. To combat the threat, emojis were added with the colors. In the experimental group, no students had issues using MOON.

Regarding external validity, we performed our controlled experiment in only one context. Therefore, the results could vary with level students or the notebook used in the session.

VI. User Study

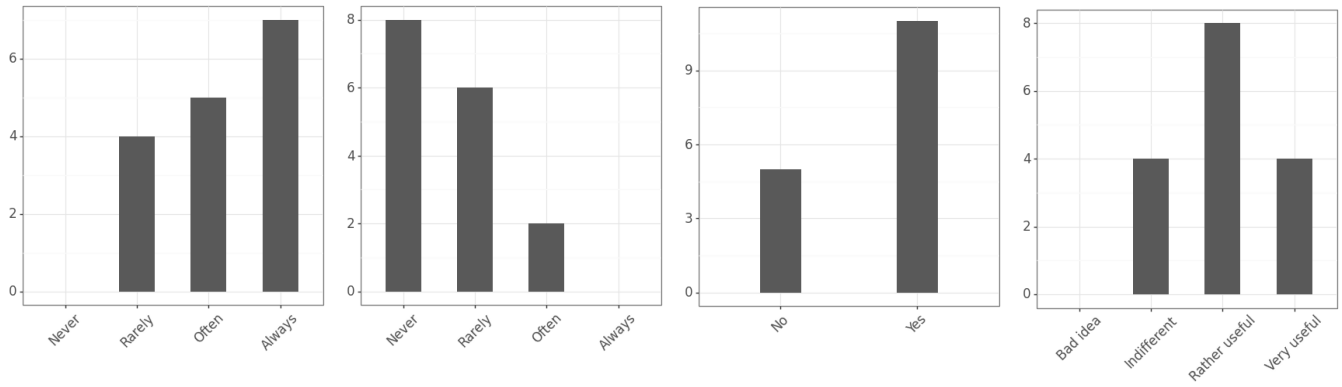
To complement our controlled experiment, we asked the 10 students involved in the experimental group and 10 other students who were involved in the beta testing of MOON to complete an anonymous online survey about MOON. We received 16 answers to this survey yielding a response rate of 80%. Figure 4 depicts the distribution of the answers to the four main questions.

To the question "*Does MOON help to follow the execution order of cells?*" we observe the distribution depicted in Figure 4a. We note that the most frequent answer is *Always*, with 7 answers. We also note that no participant answered *Never*. This result corroborates what we observed for RQ1 in the controlled experiment, where the fitness observed in the experimental group was significantly higher than in the control group.

To gain a deeper understanding of how the students perceive the actions suggested by MOON, we asked the question "*Are the actions suggested by MOON surprising?*" Figure 4b depicts the answers to this question. We note that the most frequent answer is *Never*, with 8 answers, and only two participants answered *Often*. It indicates that students mostly understand the actions suggested by MOON, but some actions might be surprising in some cases. We included a free-text question to add more context to this question. Two students mentioned that they were surprised to have to validate the intermediate cells in case of backtracking, even if these cells were not modified. One student mentioned that they were surprised to be able to execute a cell very distant from the current one.

To corroborate the results of RQ2 in the controlled experiment, we asked "*Does MOON help to progress faster?*" Figure 4c depicts the answers to this question. We note that the most frequent answer is *Yes* with 11 answers, and only 5 students answered *No*. This aligns with the fact that we did not see any effect of MOON on completeness, and we observed increased median completeness in the experimental group.

To assess whether the students found MOON helpful, we asked the question "*Do you think that MOON should be activated by default?*", *i.e.*, would the students like to use MOON in all their practical sessions? Figure 4d depicts the answers to this question. Inspired by related work [30], we decided to use an asymmetric survey response scale inspired by the Kano model suitable for evaluating preferences. We note that the most frequent answer is *Rather useful* with 8 answers, while 4 students answered *Indifferent* and 4 *Very useful*. None of the students answered *Bad idea*. It indicates that most students find MOON useful, while some do not seem



(a) Does MOON help to follow the execution order of cells? (b) Are the actions suggested by MOON surprising? (c) Does MOON help to progress faster? (d) Do you think that MOON should be activated by default?

Figure 4: Distributions of the answers for the questions of our anonymous survey

to see much benefit in using it. We conjecture this might be true for the most advanced notebook users.

Finally, we included a final free-text question asking for any feedback about MOON. One student mentioned that validating intermediate cells in case of backtracking is tedious. Another mentioned that the help provided by MOON did not stand out at first, but when he returned to the notebook without it, the difference was apparent, and the notebook seemed more complicated to use.

VII. Related work

We discuss the related work around two topics: the use of notebooks in education and digital storytelling.

The literature shows that notebooks are increasingly used in teaching[2] and that there are mainly four ways of using them: video projection (including live programming [5]), as course and exercise sheets [6, 8, 31], and to support automatic evaluation [1, 2, 3]. Our approach only encompasses the course and exercise sheets aspect which allows the student to have in the same document his code, the execution of the code and the instructions for an exercise session. But the notebook written by the teacher is often put to the test when the students are in the exploratory phase [23, 32]. It is therefore necessary for the teacher to have a precise idea of how a notebook works, its advantages but also its pitfalls. Once this knowledge has been acquired, the teacher must face a double challenge: writing instructions to guide the student in his or her learning process and acquiring good practices [33, 34] to avoid the execution pitfalls intrinsic to the notebook [26, 4].

Several tools have been developed to improve the use of notebooks in the classroom, such as plugins for automatic evaluation [1], reproducibility [16, 35], or the use of Personal Learning Environments such as GRAASP [36], with the integration of a tool resembling the notebook [37].

Several other approaches are not directly related to teaching but explore another very interesting aspect for the teacher related to storytelling by modifying the user interface of notebooks [38, 39, 40, 41, 42].

Although these tools allow for improved interactions between the notebook and the users in terms of reproducibility, sharing, and storytelling, none of them are adapted for the teacher to embody his scenario in the notebook to assist with the reading and execution of it. We propose a new approach with MOON which allows the teacher to script his scenario in an educational notebook and to activate for the students' assistance the execution of the script based on a color system.

VIII. Conclusion

In this article, we present a novel approach, MOON, designed to guide students through educational notebook scenarios. The central idea is to provide teachers with a language that enables them to formalize the expected usage of their notebooks in the form of a script and to interpret this script to guide students with visual indications in real time while they interact with the notebooks. We evaluate our approach using a randomized controlled experiment involving 21 students, which shows that MOON helps students comply better with the intended scenario without hindering their ability to progress. Our follow-up user study shows that about 75% of the surveyed students perceived MOON as rather useful or very useful. In future work, we plan to investigate how we can leverage MOON to provide feedback to teachers. A first idea would be to analyze the traces and the scripts to pinpoint pain points in a scenario and provide improvement suggestions. A second idea would be to gather data from MOON instances in real-time to directly assist the teachers during a practical session by pinpointing students struggling with the notebook or lagging in the scenario. Another important point to improve is script writing. Currently, our approach imposes to write the script for the notebook scenario manually. We are exploring ways to enhance the script writing process within the notebook scenario and ensure synchronization of the script when changes are made to the scenario.

[2]https://www.epfl.ch/education/educational-initiatives/jupyter-notebooks-for-education/

References

[1] P. Jupyter, D. Blank, D. Bourgin, A. Brown, M. Bussonnier, J. Frederic, B. Granger, T. Griffiths, J. Hamrick, K. Kelley, M. Pacer, L. Page, F. Perez, B. Ragan-Kelley, J. Suchow, and C. Willing, "nbgrader: A tool for creating and grading assignments in the jupyter notebook," *Journal of Open Source Education*, vol. 2, p. 32, 01 2019. [Online]. Available: https://doi.org/10.21105/jose.00032

[2] H. Manzoor, A. Naik, C. A. Shaffer, C. North, and S. H. Edwards, "Auto-grading jupyter notebooks," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE 2020, Portland, OR, USA, March 11-14, 2020*, J. Zhang, M. Sherriff, S. Heckman, P. A. Cutter, and A. E. Monge, Eds. ACM, 2020, pp. 1139–1144. [Online]. Available: https://doi.org/10.1145/3328778.3366947

[3] C. D. González-Carrillo, F. Restrepo-Calle, J. J. Ramírez-Echeverry, and F. A. González, "Automatic grading tool for jupyter notebooks in artificial intelligence courses," *Sustainability*, vol. 13, no. 21, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/21/12050

[4] D. H. S. IV, Q. Hao, C. D. Hundhausen, F. Jagodzinski, J. Myers-Dean, and K. Jaeger, "Towards modeling student engagement with interactive computing textbooks: An empirical study," in *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*, M. Sherriff, L. D. Merkle, P. A. Cutter, A. E. Monge, and J. Sheard, Eds. ACM, 2021, pp. 914–920. [Online]. Available: https://doi.org/10.1145/3408877.3432361

[5] C. H. Chen and P. J. Guo, "Improv: Teaching programming at scale via live coding," in *Proceedings of the Sixth ACM Conference on Learning @ Scale, L@S 2019, Chicago, IL, USA, June 24-25, 2019*. ACM, 2019, pp. 9:1–9:10. [Online]. Available: https://doi.org/10.1145/3330430.3333627

[6] C. Sutrini, D. Passaro, and F. Pallotta, "The potential of using jupyter notebook in physics education: Experimentation for high school students," *Il nuovo cimento C*, vol. 45, no. 6, pp. 1–4, 2022.

[7] A. Suárez-García, E. Arce-Fariña, M. Á. Hernández, and M. F. Gavilanes, "Teaching structural analysis theory with jupyter notebooks," *Comput. Appl. Eng. Educ.*, vol. 29, no. 5, pp. 1257–1266, 2021. [Online]. Available: https://doi.org/10.1002/cae.22383

[8] C. J. Weiss, "A creative commons textbook for teaching scientific computing to chemistry students with python and jupyter notebooks," *Journal of Chemical Education*, vol. 98, no. 2, pp. 489–494, 2021. [Online]. Available: https://doi.org/10.1021/acs.jchemed.0c01071

[9] M. van Staveren, "Integrating python into a physical chemistry lab," *Journal of Chemical Education*, vol. 99, no. 7, pp. 2604–2609, 2022. [Online]. Available: https://doi.org/10.1021/acs.jchemed.2c00193

[10] D. Lafuente, B. Cohen, G. Fiorini, A. A. García, M. Bringas, E. Morzan, and D. Onna, "A gentle introduction to machine learning for chemists: An undergraduate workshop using python notebooks for visualization, data processing, analysis, and modeling," *Journal of Chemical Education*, vol. 98, no. 9, pp. 2892–2898, 2021. [Online]. Available: https://doi.org/10.1021/acs.jchemed.1c00142

[11] E. J. Menke, "Series of jupyter notebooks using python for an analytical chemistry course," *Journal of Chemical Education*, vol. 97, no. 10, pp. 3899–3903, 2020. [Online]. Available: https://doi.org/10.1021/acs.jchemed.9b01131

[12] A. A. Smith, "Teaching computer science to biologists and chemists, using jupyter notebooks: Tutorial presentation," *J. Comput. Sci. Coll.*, vol. 32, no. 1, p. 126–128, oct 2016.

[13] J. Koehler and S. Kim, "Interactive classrooms with jupyter and python," *The Mathematics Teacher*, vol. 111, p. 304, 01 2018.

[14] G. Vial and B. Negoita, "Teaching programming to non-programmers: The case of python and jupyter notebooks," in *Proceedings of the International Conference on Information Systems - Bridging the Internet of People, Data, and Things, ICIS 2018, San Francisco, CA, USA, December 13-16, 2018*, J. Pries-Heje, S. Ram, and M. Rosemann, Eds. Association for Information Systems, 2018. [Online]. Available: https://aisel.aisnet.org/icis2018/education/Presentations/1

[15] H. Guerra, L. M. Gomes, and A. Cardoso, "Agile approach to a cs2-based course using the jupyter notebook in lab classes," in *2019 5th Experiment International Conference (exp.at'19), Funchal (Madeira Island), Portugal, June 12-14, 2019*. IEEE, 2019, pp. 177–182. [Online]. Available: https://doi.org/10.1109/EXPAT.2019.8876536

[16] C. Casseau, J. Falleri, X. Blanc, and T. Degueule, "Immediate feedback for students to solve notebook reproducibility problems in the classroom," in *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2021, St Louis, MO, USA, October 10-13, 2021*, K. J. Harms, J. Cunha, S. Oney, and C. Kelleher, Eds. IEEE, 2021, pp. 1–5. [Online]. Available: https://doi.org/10.1109/VL/HCC51201.2021.9576363

[17] K. J. O'Hara, D. S. Blank, and J. B. Marshall, "Computational notebooks for AI education," in *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2015, Hollywood, Florida, USA, May 18-20, 2015*, I. Russell and W. Eberle, Eds. AAAI Press, 2015, pp. 263–268. [Online]. Available: http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS15/paper/view/10349

[18] C. Tan, S. B. Geva, and D. Colbry, "The nascent case for adopting jupyter notebooks as a pedagogical tool for interdisciplinary humanities, social science, and arts education," *Comput. Sci. Eng.*, vol. 23,

no. 2, pp. 107–113, 2021. [Online]. Available: https://doi.org/10.1109/MCSE.2021.3062199

[19] A. Dillon, "Reading from paper versus screens: a critical review of the empirical literature," *Ergonomics*, vol. 35, no. 10, pp. 1297–1326, 1992. [Online]. Available: https://doi.org/10.1080/00140139208967394

[20] J. M. Noyes and K. J. Garland, "Computer- vs. paper-based tasks: Are they equivalent?" *Ergonomics*, vol. 51, no. 9, pp. 1352–1375, 2008, pMID: 18802819. [Online]. Available: https://doi.org/10.1080/00140130802170387

[21] L. M. Singer and P. A. Alexander, "Reading on paper and digitally: What the past decades of empirical research reveal," *Review of Educational Research*, vol. 87, no. 6, pp. 1007–1041, 2017. [Online]. Available: https://doi.org/10.3102/0034654317722961

[22] P. Delgado, C. Vargas, R. Ackerman, and L. Salmerón, "Don't throw away your printed books: A meta-analysis on the effects of reading media on reading comprehension," *Educational Research Review*, vol. 25, pp. 23–38, 2018. [Online]. Available: https://doi.org/10.1016/j.edurev.2018.09.003

[23] S. Chattopadhyay, I. Prasad, A. Z. Henley, A. Sarma, and T. Barik, "What's wrong with computational notebooks? pain points, needs, and design opportunities," in *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*, R. Bernhaupt, F. F. Mueller, D. Verweij, J. Andres, J. McGrenere, A. Cockburn, I. Avellino, A. Goguey, P. Bjøn, S. Zhao, B. P. Samson, and R. Kocielnik, Eds. ACM, 2020, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3313831.3376729

[24] A. Head, F. Hohman, T. Barik, S. M. Drucker, and R. DeLine, "Managing messes in computational notebooks," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*, S. A. Brewster, G. Fitzpatrick, A. L. Cox, and V. Kostakos, Eds. ACM, 2019, p. 270. [Online]. Available: https://doi.org/10.1145/3290605.3300500

[25] M. B. Kery, M. Radensky, M. Arya, B. E. John, and B. A. Myers, "The story in the notebook: Exploratory data science using a literate programming tool," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, R. L. Mandryk, M. Hancock, M. Perry, and A. L. Cox, Eds. ACM, 2018, p. 174. [Online]. Available: https://doi.org/10.1145/3173574.3173748

[26] A. Rule, A. Tabard, and J. D. Hollan, "Exploration and explanation in computational notebooks," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, R. L. Mandryk, M. Hancock, M. Perry, and A. L. Cox, Eds. ACM, 2018, p. 32. [Online]. Available: https://doi.org/10.1145/3173574.3173606

[27] C. Casseau, J.-R. Falleri, T. Degueule, and X. Blanc, "MOON: Assisting Students in Completing Educational Notebook Scenarios (Artifacts)," July 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8167588

[28] D. S. McNamara and J. Magliano, "Chapter 9 toward a comprehensive model of comprehension," in *The Psychology of Learning and Motivation*, ser. Psychology of Learning and Motivation. Academic Press, 2009, vol. 51, pp. 297–384. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079742109510092

[29] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation, 3rd Edition*, ser. Pearson international edition. Addison-Wesley, 2007.

[30] A. Begel and T. Zimmermann, "Analyze this! 145 questions for data scientists in software engineering," in *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, P. Jalote, L. C. Briand, and A. van der Hoek, Eds. ACM, 2014, pp. 12–23. [Online]. Available: https://doi.org/10.1145/2568225.2568233

[31] F. M. Sallabi and S. Lazarova-Molnar, "Teaching modeling, simulation, and performance evaluation course online with jupyter notebook: Course development and lessons learned," in *IEEE Frontiers in Education Conference, FIE 2022, Uppsala, Sweden, October 8-11, 2022*. IEEE, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1109/FIE56618.2022.9962690

[32] J. W. Johnson, "Benefits and pitfalls of jupyter notebooks in the classroom," in *SIGITE'20: The 21st Annual Conference on Information Technology Education, Virtual Event, USA, October 7-9, 2020*, D. Khazanchi, H. P. Siy, G. Grispos, and T. K. Setor, Eds. ACM, 2020, pp. 32–37. [Online]. Available: https://doi.org/10.1145/3368308.3415397

[33] J. Wang, L. Li, and A. Zeller, "Better code, better sharing: on the need of analyzing jupyter notebooks," in *ICSE-NIER 2020: 42nd International Conference on Software Engineering, New Ideas and Emerging Results, Seoul, South Korea, 27 June - 19 July, 2020*, G. Rothermel and D. Bae, Eds. ACM, 2020, pp. 53–56. [Online]. Available: https://doi.org/10.1145/3377816.3381724

[34] L. Quaranta, F. Calefato, and F. Lanubile, "Eliciting best practices for collaboration with computational notebooks," *Proc. ACM Hum. Comput. Interact.*, vol. 6, no. CSCW1, pp. 87:1–87:41, 2022. [Online]. Available: https://doi.org/10.1145/3512934

[35] H. Fangohr, V. Fauske, T. Kluyver, M. Albert, O. Laslett, D. Cortés-Ortuño, M. Beg, and M. Ragan-Kelly, "Testing with jupyter notebooks: Notebook validation (nbval) plug-in for pytest," *CoRR*, vol. abs/2001.04808, 2020. [Online]. Available: https://arxiv.org/abs/2001.04808

[36] D. Gillet, A. Vozniuk, M. Rodríguez-Triana, and A. Holzer, "Agile, versatile, and comprehensive social media platform for creating, sharing, exploiting, and archiving personal learning spaces, artifacts, and traces,"

in *Proceedings of the 6th World Engineering Education Forum*, 11 2016.

[37] A. D. Santo, J. C. Farah, M. L. Martínez, A. Moro, K. Bergram, A. K. Purohit, P. Felber, D. Gillet, and A. Holzer, "Promoting computational thinking skills in non-computer-science students: Gamifying computational notebooks to increase student engagement," *IEEE Trans. Learn. Technol.*, vol. 15, no. 3, pp. 392–405, 2022. [Online]. Available: https://doi.org/10.1109/TLT.2022.3180588

[38] H. Li, L. Ying, H. Zhang, Y. Wu, H. Qu, and Y. Wang, "Notable: On-the-fly assistant for data storytelling in computational notebooks," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, A. Schmidt, K. Väänänen, T. Goyal, P. O. Kristensson, A. Peters, S. Mueller, J. R. Williamson, and M. L. Wilson, Eds. ACM, 2023, pp. 173:1–173:16. [Online]. Available: https://doi.org/10.1145/3544548.3580965

[39] D. Kang, T. Ho, N. Marquardt, B. Mutlu, and A. Bianchi, "Toonnote: Improving communication in computational notebooks using interactive data comics," in *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021*, Y. Kitamura, A. Quigley, K. Isbister, T. Igarashi, P. Bjørn, and S. M. Drucker, Eds. ACM, 2021, pp. 727:1–727:14. [Online]. Available: https://doi.org/10.1145/3411764.3445434

[40] N. Weinman, S. M. Drucker, T. Barik, and R. DeLine, "Fork it: Supporting stateful alternatives in computational notebooks," in *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021*, Y. Kitamura, A. Quigley, K. Isbister, T. Igarashi, P. Bjørn, and S. M. Drucker, Eds. ACM, 2021, pp. 307:1–307:12. [Online]. Available: https://doi.org/10.1145/3411764.3445527

[41] J. Harden, E. Christman, N. Kirshenbaum, J. Wenskovitch, J. Leigh, and C. North, "Exploring organization of computational notebook cells in 2d space," in *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2022, pp. 1–6.

[42] Z. J. Wang, K. Dai, and W. K. Edwards, "Stickyland: Breaking the linear presentation of computational notebooks," in *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022, Extended Abstracts*, S. D. J. Barbosa, C. Lampe, C. Appert, and D. A. Shamma, Eds. ACM, 2022, pp. 269:1–269:7. [Online]. Available: https://doi.org/10.1145/3491101.3519653