European Journal of Physics

Integrating Python data analysis in an existing introductory laboratory course

OPEN ACCESS
IOP Publishing
European Journal of Physics
Eur. J. Phys. **45** (2024) 045707 (18pp)
https://doi.org/10.1088/1361-6404/ad4fcc

Integrating Python data analysis in an existing introductory laboratory course

Eugenio Tufino[1,*], **Stefano Oss**[1] and **Micol Alemani**[2,*]

[1] Department of Physics, University of Trento, I-38123, Trento, Italy
[2] Institute of Physics and Astronomy, University of Potsdam, D-14476, Potsdam, Germany

E-mail: eugenio.tufino@unitn.it and alemani@uni-potsdam.de

Received 17 September 2023, revised 26 March 2024
Accepted for publication 23 May 2024
Published 17 June 2024

Abstract

In this article we describe how we successfully incorporated data analysis in Python in a first year laboratory course without significantly altering the course structure and without overburdening students. We show how we created and used carefully designed Jupyter Notebooks with exercises and physics application examples that allow students to master data analysis programming in the laboratory course. We use these Notebooks to guide students through the fundamentals of data handling and analysis in Python while performing simple experiments. We present our teaching approach and the developed materials. We discuss the effectiveness of our intervention based on the results from pre- and post-course questionnaires and students' group work. The results presented give insights about advantages and challenges of introducing computation at the early stage of the curriculum in a laboratory course setting and are informative for other instructors and the physics education research community.

Supplementary material for this article is available online

Keywords: data analysis, physics education, laboratory courses, computational skills, Jupyter Notebooks, Python

* Authors to whom any correspondence should be addressed

Introduction

In the fast-paced era of technological advancement, computational skills have become a necessity across various disciplines, including physics. According to the World Economic Forum's projections for 2025 [1], there is a quickly rising demand in the job market for individuals who can harness the power of computation to analyse complex data and create innovative solutions to problems.

Also, in all fields of experimental science, the rate at which data is being generated is accelerating, and the use of robust tools for data analysis and interpretation has become a necessity. Computational skills are increasingly demanded especially in large-scale projects in particle physics or astrophysics, where both the management and analysis of big data are essential.

Due to these demands, there is a need to teach 'computational skills' in the physics curriculum, a step beyond the conventional use of spreadsheets and integrated mathematical computing packages typically found in physics laboratory courses (PLCs). By fostering these competencies, students are not only prepared for their future careers in areas that increasingly require these skills but can also enrich their understanding of the studied subjects. Introducing computation into high school or undergraduate courses has recently become an active area of physics education research. It is now widely recognized that in the practice of physics, there are three fundamental elements: experimentation, theory, and computation. This interest in integrating computation has led to specific recommendations from the American Association of Physics Teachers (AAPT) [2–4], and from the Partnership for Integration of Computation into Undergraduate Physics (PICUP) group [5], from which abundant material can be sourced. One can find several works by the physics education research community in the past years that aim to introduce computation into diverse physics curriculum settings. Examples include the integration of computation into existing physics lectures courses [6–8], the design of project-based computational courses [9] and the integration of computation into laboratory settings [10–12]. For example, for introductory first year laboratory courses, researchers used Microsoft Excel spreadsheets [10] or VPython modules [11–13] while for more advanced laboratory settings programming languages like Python [10] were utilised. For a comprehensive and updated (2023) overview on incorporating computation into physics education, see also the Resource Letter [14].

Our work aims to contribute to these efforts and explore how to introduce data analysis in Python already in a first semester laboratory course. We choose Python because of its wide use by experimental physicists in laboratory settings, its appeal for the job market, its flexibility, and its extensive documentation searchable on the web. Our approach was motivated by our ongoing efforts in creating a teaching environment in our laboratory course in which students are introduced to authentic scientific practices from the very beginning. We believe that introductory PLCs offer a special and desirable environment for introducing programming skills early in the curriculum. In fact, in these settings students have many opportunities to practise programming data analysis with their own data, be active and get frequent feedback. This is facilitated by the fact that students work in small groups and are supported by a small ratio of students-instructors. Starting at the early stage of the curriculum it's important to provide all students with the opportunity to engage and get practice with computation and avoid that in more advanced courses with more sophisticated use of computation only students with previous programming experience participate and succeed. Most importantly since computational work in Python is an actual part of experimental practices by physicists, it is desirable to include it from the beginning into a lab course where authentic practices are learned.

Our task is however not simple and special care must be adopted to successfully integrate programming computational tools into these settings. In fact, in addition to the challenges that have been already discussed in the literature for introducing computation into the physics curriculum [15], the specific context of a first semester laboratory course presents a few more. These courses are in fact already dense with learning goals, for example being able to apply statistical analysis techniques, deal with measurement uncertainties or document experiments [16]. Additionally, first semester students might still be adapting to their new higher education environment.

In this paper we describe how we introduced data analysis in Python into a first semester introductory course by lowering the barrier towards programming and allowing students to be operative doing their data analysis in Python from the very first experiments.

Our teaching approach, the definition of computational learning goals, the developed curriculum and the effect of the intervention on students' learning and beliefs are presented and discussed in the paper. To evaluate the intervention, we used a multi-faceted and prolonged approach that aimed to investigate the achievement of the learning goals. The discussion of our assessment contributes to the recent efforts of the physics education research community to discuss and create forms of assessments for evaluating the acquisition of computational skills and/or the investigation of students' attitudes and/or dispositions towards computation [17].

Design and implementation of the intervention

The design and implementation of this intervention consisted of three steps. First, we set learning goals regarding the introduction of data analysis in Python in the first semester laboratory course. We refer to those goals as *laboratory computational learning goals* (see section below '*Laboratory Computational Learning Goals*') as they are a subset of learning goals that we added for this intervention. Secondly, we developed curricular materials and activities to foster students' achievement of those computational learning goals and to assess them. Third, we evaluated the achievement of the computational learning goals by analysing students' work during the different types of activities and by investigating students' answers to pre-post questionnaires.

In the following sections we describe in detail the different steps of this process, i.e. setting the learning goals, the design of the computational activities and their assessment and also the context in which the intervention took place.

Existing course structure and learning goals

We implemented the newly designed introduction to Python data analysis in the academic year 2022–2023 of the first-semester introductory laboratory course for physics major students at the University of Potsdam in Germany. This course is the first of a sequence of obligatory modules of the PLC taking place during the first four semesters of the curriculum. These modules have been redesigned starting from 2016 by one of the authors (M A) focusing on students' development of experimental skills such as design, modelling, communication, and technical skills [18]. A further goal is to have students understand the nature of experimental physics and learn to 'think like a scientist' [19]. The teaching approach abandons traditional cookbook-style experiments, in favour of a more student-centred authentic approach. One notable feature includes the use of laboratory notebooks rather than conventional laboratory reports, to promote a more engaged, authentic, and reflective learning experience [20, 21]. Communication and collaboration are fostered through carefully

designed activities conducted in groups. Our introductory laboratory course sequence at University of Potsdam is so designed that for each semester we focus on the achievement of a specific set of learning goals. For the first semester course the already existing learning goals are: (i) students are able to apply the basic concepts of measurement uncertainties and systematic errors; (ii) students are able to write a good laboratory notebook and explain why physicists use laboratory notebooks; and (iii) students are able to recognize what components make up a good graph, to create a good graph, and to use graphical analysis methods (for example, least squares fitting and linearization). The course in the first semester includes both a seminar component with exercises, group work, quizzes, peer instruction, and homework problems and a laboratory component. In the laboratory component, students do simple experiments to practise working with measurement uncertainties, systematic errors, graphing, and writing authentic laboratory notebooks, therefore applying the skills trained during the seminar component to the laboratory component.

In the newly introduced intervention on computation that we are describing in this paper, we kept the above described existing laboratory goals and activities of the course almost unchanged apart that we reduced the amount of in-class exercises and homework problems during the seminar component of the course and substituted the introduction to the use of the open source spreadsheet-type software SciDavis [22] for visualising and analysing data with the introduction to Python. Therefore, before the intervention on computation described here, students had already received instruction and practised on how to evaluate measurement uncertainties, how to conduct graphical analysis, and how to maintain a laboratory notebook.

We notice that since the assessment of the above described already existing learning goals in our laboratory course was already performed earlier elsewhere [18, 19]. We will concentrate in the rest of this paper only on the definition and assessment of the newly formulated learning goals on computation.

Laboratory computational learning goals

On top of the above described already existing learning goals of our laboratory course, we introduced new learning goals on the use of Python data analysis in the first semester.

For this aim, we utilised the AAPT recommendations for computational physics in the undergraduate physics curriculum [2] and we reviewed the recent literature on the topic, such as Weller, with his framework on Computational Thinking (CT) practices [23], and the works of Odden and Sorrenson [7] and Caballero and Pollock [8].

We defined the following *laboratory computational learning goals*, which we divided into two categories: *Implement* and *Communicate*. We list the learning goals related to these categories here:

Implement:

- Students are able to write Python codes to manipulate, analyse, and visualise data sets.
- Students are able to write Python codes to implement computational methods to find solutions.

Communicate:

- Students are able to communicate clearly their work using Jupyter Notebooks (JN), being able to integrate images, texts, and codes in a JN.
- Students are able to write clear, well-commented Python codes.
- Students can create clear, informative and complete graphs using Python.

These broad learning goals are then translated into more detailed learning objectives, tailored to individual lessons.

As one can notice, our emphasis is on very specific and introductory level computational skills for data analysis, not on teaching general coding skills.

It is important to note that the ability to identify relevant and appropriate features in a physics problem or model, as well as to identify the appropriate data analysis techniques, is considered in our study a prerequisite skill. We monitored this skill, but this analysis is not included in this paper because we want to focus specifically on computational skills.

A further goal pursued during our intervention is to foster positive attitudes of students towards computational work and enhance students' self-efficacy and motivation in this domain. In fact, when people believe they are capable of completing a task, a concept referred in the literature as self-efficacy, they are more likely to engage and persist in the task [24].

Assessment of the laboratory computational learning goals

The *computational learning goals* newly introduced into our laboratory course have been evaluated through both formative and summative forms of assessments. In the first two lessons dedicated to Python programming, students were required to study and complete the four provided JNs containing examples and exercises (see section 'Methods and Data Collection' for details).

Then, students had to apply the concepts and material of the previous lessons for doing data analysis into the two following experimental activities.

In the following semester, an in-class test consisting of two exercises was conducted and the use of computation in the experimental activities of the second semester was monitored.

To evaluate students' work in the two initial lessons dedicated to Python, in the two laboratory activities in the first semester and in the in-class test in the second semester, we have developed an evaluation rubric with categories aligned with our learning goals in the categories *Implement and Communicate*. In the rubric, we used for each category scores ranging from 0 (missing) to 3 (near mastery).

On the other hand, for the assessment of our learning goals regarding students' attitudes towards computation we made use of a pre-post survey conducted during the first semester (see section 'Methods and Data Collection').

Curriculum design: integrating computation with Jupyter Notebook in the lab course

To introduce Python programming language within the introductory PLC we utilised Jupyter Notebook, an open-source web-browser-based interactive application that facilitates the creation and sharing of documents containing code, equations, figures, and text [25]. We developed several introductory JNs to teach the fundamentals of Python in presence, actively engaging students in PLCs in a collaborative learning process. Additionally, we prepared more JNs, with examples of physics applications and data handling, that students can use independently and asynchronously. Our approach to computational instruction consisted in a gradual increase in difficulty, from the basics of Python programming (for example doing simple calculations and text formatting) to more advanced data-handling and analysis techniques. While developing the material, we made sure that no prior coding experience was required. Since we were able to base our data analysis programming activities on existing content knowledge on graphical analysis and measurement uncertainties, our JNs were designed to directly engage students in solving coding exercises and problems in groups. Examples of these exercises included calculating the mean and standard deviation, plotting data with error bars, performing linear regression, computing statistical quantities like χ^2 and

(a)

Introducing the moving average

designed and developed by Eugenio Tufino and Micol Alemani

We apply the moving average technique to the graph showing the number of sunspots of our Sun as a function of time.

This method is useful to smooth a noisy curve. For more details see here: moving average on wikipedia.
This exercise is inspired by the excellent book 'Computational Physics' by Newman.

```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
```

(b)

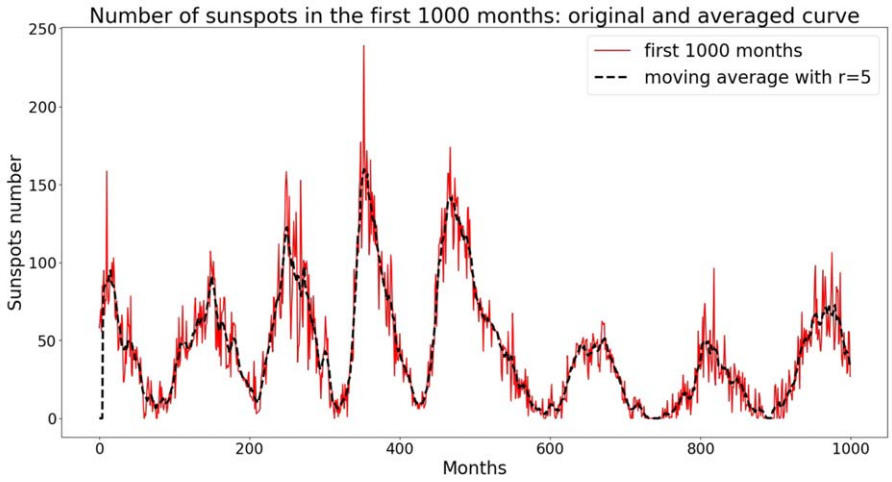

Figure 1. (a) Snapshot of one of our Jupyter Notebooks containing text, code and images. The goal of this notebook is to introduce the use of the moving average for smoothing noisy curves. An example of the use of the moving average is shown in (b) for the number of sunspots as a function of time.

r^2, and creating histograms. We made extensive use of well-known scientific libraries such as Numpy, SciPy, and Pandas, integral to modern scientific computing.

To motivate students in engaging further with data analysis in Python, we provided them also with real-world scientific applications. For example, we showed how to import a data file and calculate the moving average for smoothing the data of the number of sunspots in time (see figure 1).

This approach was used to allow students to be quickly operative in doing their data analysis during their subsequent laboratory experiments.

The JNs here described are accessible on GitHub (in the English and German versions) for broader educational use [26].

Instructors of similar courses are welcome to utilise these JNs, which cover standard topics for an introductory physics lab course. While the examples and practical scientific applications provided in our notebooks offer comprehensive perspectives, instructors can adapt the content to fit the specific requirements and objectives of their courses, for example, by introducing examples from biology, engineering, and other disciplines.

We notice here that JNs can be accessed through cloud platforms like Google Colab [27], allowing for cloud usage without software installation. However, due to privacy constraints in Germany, we were unable to fully exploit this cloud-based feature and instead worked within

Table 1. Python data analysis curriculum implemented in the introductory laboratory course.

Notebooks's material	Short description	Type of instruction
Getting started with Python (four JNs)	First laboratory session: introduction to various JN platforms and their installation. Basic calculations, formatting text, Latex, basic plot Second laboratory session: introduction of scientific libraries, use of Pandas dataframe, linear fit, statistical quantities like χ^2 and r^2, and histograms	Two in-person lab sessions, three hours each
Application examples (five JNs)	Implementing Python to analyse real-world physics phenomena: calculating the moving average applied to sunspot time-series, plotting and analysing light source illuminance versus distance, creating histograms of the period of oscillation of a pendulum, performing a simulation of the parabolic motion, and plotting atmospheric CO_2 data from website	Asynchronous work, to be done independently for the next lab module
Additional techniques (two JNs)	Techniques for importing and manipulating data files across various platforms like Jupyter Lab-Anaconda, Google Colab, and the Internet	Asynchronous work, to be done independently for the next lab module

the Anaconda environment in the classroom [28]. Even if the Anaconda environment requires students to go through installation processes, we did not experience problems with this aspect.

In two dedicated laboratory sessions, we introduced the foundations of Python for basic data analysis using four JNs (see table 1). Additional JNs were provided for students to work asynchronously on application examples to further deepen their skills. In table 1, we describe the details of the intervention.

Methods and data collection

To evaluate the impact on students' beliefs of our intervention, we administered pre- and post-course questionnaires in German. The pre-survey served as a valuable tool in assessing students' previous experience with programming and their views on computational skills in general, allowing the instructors to tailor the course content and approach accordingly. The post-survey provided us with students' feedback on the intervention.

In addition to the questionnaires, we have also evaluated hands-on exercises and have performed an in-depth analysis of students' work.

This multi-faced methodology allowed us to understand students' prior competencies, their engagement with the computational components, and their reception of the new methods, as well as to assess the development of their skills throughout the course.

Participants

48 students participated voluntarily in this study. During the course students were supervised by teaching assistants (TAs), and by authors of the study, one of which is also the course coordinator. Students worked typically in groups of three. To form a group, students had the option to choose their own group members. If they did not express a preference, they were assigned by the instructor. For the assessment, students submitted their group-work. To foster individual learning and skill sharing within a group, we promoted the rotation of roles among team members, such as who would write the code for data analysis and documentation, who would document the experiment's progress, and who would handle the experimental apparatus.

Pre- and post-surveys

The questionnaires were designed and validated by our team and filled out by students in a paper format. All participants were informed of the study's purpose and participated voluntarily. To be able to match students' answers to the questionnaires, students were asked to self-generate an anonymous code. The pre- and -post surveys were compiled by students individually. We obtained 42 matched answers between the pre-course and post-course surveys.

The pre-course survey consisted of 11 questions (see supplementary material for the surveys) and was designed to investigate students' initial background and familiarity with programming. It also sought to understand students' expectations and attitudes towards the integration of programming into the first semester PLC. The post-course survey consisted of an expanded set of questions. To be able to do a comparison, some of the pre-survey questions on students' expectations regarding programming were repurposed in the post-survey. In the post-survey students were also given the opportunity to articulate their thoughts through open-ended answers, express their judgments on specific aspects of the Python module, and provide feedback on group work. The inclusion of open-ended questions allowed for a more in-depth understanding of students' engagement in the course.

In the analysis of the open-ended responses, we employed an inductive methodology, utilising coding techniques to identify patterns and themes [29].

Analysis of students' work during the first semester

We examined how students tackled in groups the exercises given during the two lessons in the first semester fully dedicated to Python. Furthermore, we analysed laboratory notebooks of student groups during the two subsequent experiments of the first semester, specifically looking at how they utilised Python and the JNs. This analysis provided insight into the students' practical application of programming skills and the integration of computational methods into their experimental work.

Analysis of students' work during the following (second) semester

At the beginning of the second semester, we let student groups solve two in class exercises. These were specifically designed to challenge students to apply the techniques they had learned during the previous semesters and to reinforce their understanding of the programming concepts. A description of these exercises is in table 2.

Further insights were gained from observing how well students used Python for data analysis and graphing techniques in their second semester experiments.

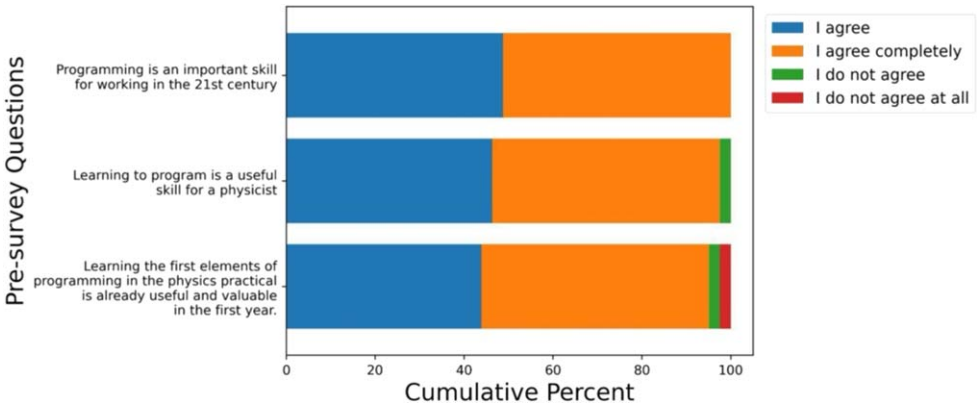

Figure 2. Stacked bar plots of students' answers to pre-survey questions about the importance of programming.

Table 2. Description of the two exercises assigned to student groups at the start of second semester.

Exercise	Objectives	Methods to use
Exercise 1: methane annual increase	Analyse global atmospheric methane increase using data from the Global Monitoring Laboratory website	Import data from webpage. Plot the data and uncertainties
Exercise 2: is the Stefan–Boltzmann law applicable to a light bulb?	Students are tasked to investigate whether the Stefan–Boltzmann law holds true for a light bulb by analysing measured data	Import voltage, current, and temperature (T) data from a provided Excel file. Plot electric power versus T^4. Then, perform linear regression and calculate r^2

Results and discussions

Pre-survey findings

From the pre-survey, we gathered information about students' previous experience in programming and on their views and expectations on learning how to do data analysis in Python in the PLC. Regarding previous programming abilities, 62% of the students already had some experience (ranging from a few weeks to several years), with 38% of those students being somehow familiar with Python and only a few (7%) having extensive experience with JNs.

We also obtained important insights regarding students' views on learning programming: students unanimously recognize the importance of learning programming at an early stage of their undergraduate education and for their careers.

For detailed response percentages to questions about the significance of programming in the 21st century, its relevance to physicists, and the value of introducing programming in the first-year physics lab, please refer to the stacked bar chart in figure 2.

An additional item in the pre-survey asked students about their expectations regarding the feasibility of learning programming in the physics lab. The majority of the students (about

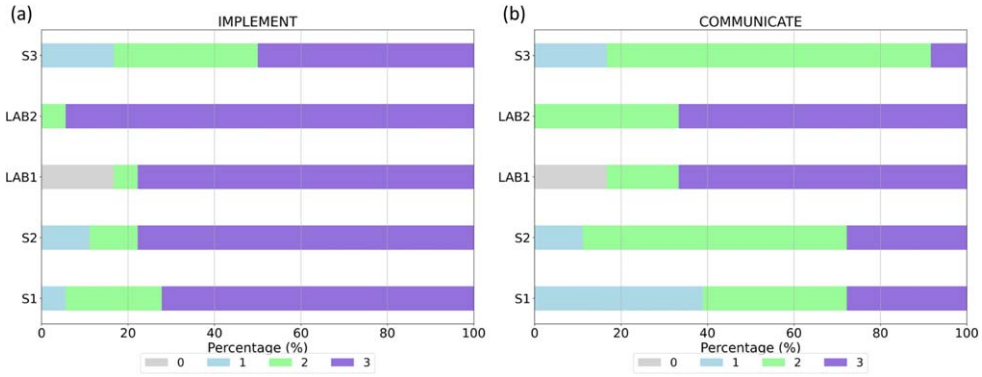

Figure 3. Percentage distribution of student scores in the Jupyter Notebooks (JNs) across different course activities, categorized under *Implement* (a) and *Communicate* (b). The initial Python programming sessions are labelled as S1 and S2. The first semester's laboratory experiments are denoted as LAB1 and LAB2, and the in-class exercises conducted at the start of the second semester are marked as S3. The scores are: 0 = missing, 1 = not adequate, 2 = needs some improvement, 3 = near mastery.

73.8%) expect the course to be feasible, about 16.7% expect that it will be easy or very easy, while only 9.5% anticipate it to be difficult.

In our opinion, these findings show that students have very positive attitudes towards programming. This result can potentially encourage other instructors to include computational aspects in their PLC, since students' epistemology on a certain subject clearly influences their learning [30]. Moreover, instructors do not have to overcome possible students' resistances to the introduction of new subjects.

Analysis of students' work during the first semester

In the introductory sessions dedicated to Python, students were guided through various exercises contained in the JNs we provided (see table 1 and [26] for details). Overall, the sessions went smoothly. While some groups finished well ahead of schedule, others completed the exercises within the allotted time. At the end of each session, student groups were required to submit their completed JNs.

In figure 3 we report the results of the assessment of students' JNs during the first two sessions dedicated to introducing Python (indicated as S1 and S2). For this, we used the categories *Implement* and *Communicate* of the rubric as described in the section '*Assessment of the laboratory computational learning goals*. We found that in the first two sessions the majority of students were able to effectively work on the exercises as shown by the prevalence of high scores, particularly in skills associated with implementation.

In the two laboratory sessions following the two introductory sessions dedicated to Python, students worked in groups on two experiments. In the first experiment, inspired by Holmes [31], students brought elastic objects from home to explore whether they obey Hooke's law. In the second experiment, students worked with a pendulum.

Examples of the student group's works using Python during the first experiment are shown in figure 4. In the first example (see figure 4(a)) students studied the behaviour of an elastic band brought from home, attempting using Python to determine which fit was most suitable to describe the data. At the end of this laboratory session, students had also the opportunity to freely develop their own 'research question' on the studied system. For instance, some groups

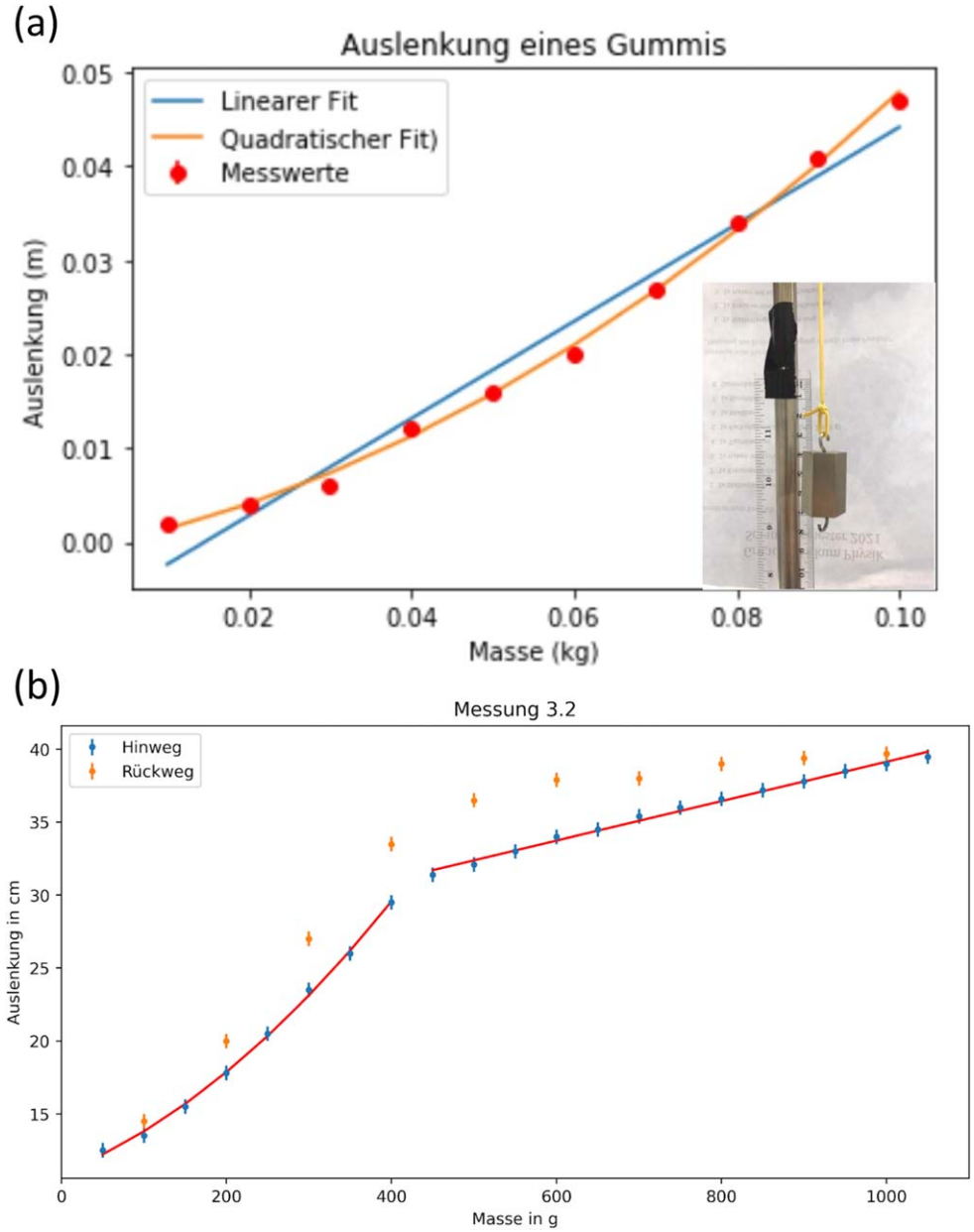

Figure 4. Extracts from laboratory notebooks of two student groups. (a) Graphical analysis in Python of the dependence between a rubber band elongation (in German 'Auslenkung') and the attached mass (in German 'Masse') with linear and quadratic fit. In the inset is a picture of the experimental setup. (b) Another plot of data obtained with a set-up as in (a) in which students analyse the behaviour of a rubber band near its breaking point (see text).

studied what happens as the elastic band approaches its breaking point. An example of such enquiry can be seen in figure 4(b). After breaking a couple of thin rubber bands, this group of students used a thicker rubber band and attached several weights on it. By doing a graphical

analysis in Python of the rubber band elongation as a function of the attached mass, they realised the existence of two regimes in the graph. They therefore did two different fits of the data. Notice that in their graph, they also showed (in orange) the data obtained by removing the weights from the rubber band, indicating the existence of a hysteresis in the system.

As exemplary shown in figure 4, during our examination of students' laboratory notebooks of these two experiments, we observed a marked increase in the utilisation of Python and JNs. In figure 3 the analysis of students' work using the rubric is also shown (indicated as LAB1 and LAB2). We found that students were able to implement their data analysis in Python and scored well in both categories of *Implement* and *Communicate*. An improvement is observed in both categories from the first to the second laboratory activity. Even if the use of Python and JN was not mandatory, 83% of student groups used Python and JNs for their data analysis in the first experiment and 100% in the second experiment.

These results indicate an overall increase in students' confidence and proficiency in using these tools. We also found that while students consider JNs effective for analysis and visualisation, they found them less convenient for taking real-time notes during the experimental process. Many (28% of the groups for both experiments) used the JNs only for the data analysis and preferred to hand-write the experimental details and documentation on tablets, paper, or use word processing software like Word, rather than recording them directly within the JNs.

Interestingly we observed an unexpected positive effect on students' attitudes during experimentation that might be related to the introduction of computation. We experienced a self-driven engagement of students in doing data analysis while (or immediately after) taking experimental data and not postponing the data analysis to the end of (or after) the lab. In past courses, we repeatedly encountered the attitude of postponing data analysis and had to openly discourage it in our PLC. Expert experimental physicists in fact use preliminary data analysis 'on the spot' to be able to progressively make decisions in the laboratory based on their experimental results and promptly recognise problems. Even if further effort must be spent to investigate this qualitative observation, we can speculate here that introducing computational aspects for the data analysis might make students want to engage in it more actively and promptly during the laboratory. They might consider data analysis as an integral part of an experimental challenge and therefore become more motivated in doing it immediately. As a positive possible consequence, they might also experience the importance of data analysis for decision making when they for example find unexpected results from their analysis and obtain more 'expert-like' attitudes toward experimental physics.

Post survey results

It is remarkable that students' already positive views on the importance of computational skills not only remained consistent but even improved between the pre-survey and the post-survey (see figure 2). It is well known by physics education research, such consistency is not a given, as students often approach new subjects with initial beliefs that may or may not align with their views after instruction.

By analysing the open-ended answers to the question 'Why is it important to learn the first elements of programming in the physics lab course already in the first year?', we found that students perceived the utility of programming for data analysis in different settings: in the laboratory, in subsequent physics courses, and in their future workplace. They recognise the importance of an early exposure to these skills because it allows for more practice. When asked about the advantages of programming data analysis in Python, students mentioned the potential time-saving benefits through automation and the general applicability in subsequent

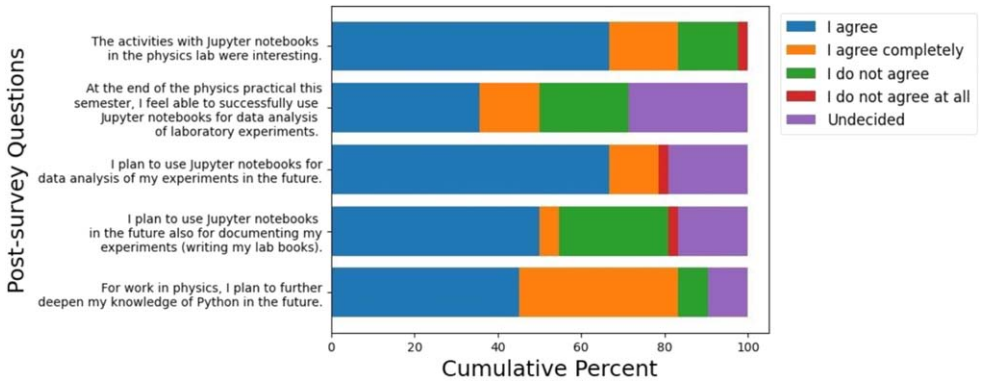

Figure 5. Stacked bar plots representing students' responses to five key questions of the post survey[3].

physics courses and professional work. We believe that these students' responses collectively reflect a future-oriented mindset.

Students' evaluation on the course, revealed that most students (65%) found the introduction to Python feasible, 17.5% believed it was 'easy' or 'very easy' and 7.5% of students that it was 'difficult'. Notably, no students chose 'Very difficult'.

Those who found the introduction to Python 'easy' or 'very easy' generally had prior programming experience and appreciated the clear instructional structure. In the 'feasible' category, 14 out of 25 students reported their feeling of a growth in their learning.

Examples of students' quotes (translated from German) explaining this growth are: '*if you have never worked with Jupyter Notebook before it is a bit complicated i.e. lots of new information. You have to develop an understanding of it first then it is easy*' and '*there were slight difficulties but if I tried hard, it was doable*'.

Among the students that reported that the introduction to Jupyter was 'difficult', some cited the high workload in the first year due to mathematics courses and physics, some reported their lack of prior experience as a challenge, and few complained about too much content and too little time.

Since the large majority of the students (82.5%) did not experience difficulties during the course and nobody experienced the Python introduction as very difficult, the fact that 17.5% of the students encountered some challenges can be considered as normal in an educational setting. To further assist the students who found computing challenging, instructors could include a third in-class session, if feasible.

The answers to further post-survey questions (see figure 5) allow for deeper investigation, revealing the following trends: a significant majority of students (84%) found the activities with JNs interesting, with 78% of students indicating plans to use JNs in the future and 86% plans to further deepen their skills in Python. However, we found that only 50% of the respondents felt capable of successfully using JNs for data analysis, while 29% were undecided or did not feel capable (21%).

To better understand, we cross-referenced the responses from the seven students who reported difficulties with their answers to the questions in figure 5. We notice that despite the difficulties these 7 students did not feel overwhelmed and remained motivated, in fact most of them plan to further use JNs and want to deepen their skills in Python.

Interestingly, the survey responses from individual students partially contrast with our analysis of students' group work that shows all groups capable of successfully programming

data analysis in Python, as discussed in the previous section. This discrepancy could potentially indicate that students are underestimating their own capabilities but could also be a result of group dynamics. To further examine this aspect, two additional questions in the post-survey were especially insightful. When asked, 'How satisfied are you with the group work during the Jupyter Notebooks activities?' the majority (71%) of students expressed satisfaction, while 12% were unsatisfied and another 12% neutral.

Examples of positive comments in the other (open ended) question are in the following (translated from German):

'*Was fun and effective too*,' '*It was fun and educational*', '*Worked well, everyone contributed to the task, was much more fun in the group*,' '*We harmonised well as a group and everyone had their tasks*,' '*6 eyes see more than 2,' 'You could always ask group members for help. I could not have done it alone', 'Because we worked very well together in the group*'.

These results show that teamwork was a strength in the trial, leading to good results. On the other hand, students unsatisfied by group work mentioned:

'All group members were at different levels, difficult distribution of tasks', 'I had programming experience, had to wait', 'The division of labour was not balanced, there were often disagreements', 'Due to a group member, one had to take over the work for two people', 'The work often stuck to one person.'

This shows that targeted strategies on group work are needed in later iterations of the course, to balance groups equally, to make even more students comfortable.

The lack of confidence in using Python expressed by 21% of the students during the post-survey is not surprising. In fact, the post-survey was proposed after only four weeks of practice since the intervention began. As it is well known, the development of scientific skills takes several weeks. For example, in the case of scientific abilities in the ISLE environment, a period of five to eight weeks was measured [32].

To account for this, additional practice opportunities should be provided. In the following section we explain how we implemented this in our intervention.

Students' work in the following semester

At the beginning of the second semester, we gave students two exercises as review to Python programming (see section 'Methods and Data Collection'). The analysis of students' group work using the rubric on computational laboratory goals is shown in figure 3 as S3. It indicates a satisfactory students' mastery of the skills and techniques introduced through the JNs in the previous semester. In fact, approximately 83% of the group achieved a score of 2 or 3 in both the *Implement* and *Communicate*. For only a few groups the work was evaluated as not adequate (score 1). It is important to note that these exercises were assigned after a three-month break between the two laboratory courses. This interval, along with the different types of exercises, could partially explain the lower scores compared to earlier sessions.

Additionally, considering students' group work during one experiment of the second semester we found that 100% of the groups used Python to produce their graphs, but only 19% of the groups used JNs as laboratory notebooks. The other 81% embedded the graphs produced in Python with the JNs in other types of software for the documentation of their work. An example of groups' work with an experiment focused on Brownian motion (see

[3] In the original German version of the post-questionnaire the term used is 'interessant' which has a positive connotation. In English speaking classroom settings, one could ask if the activities have been 'fun and interesting' in order to clearly ask if the experience was positive.

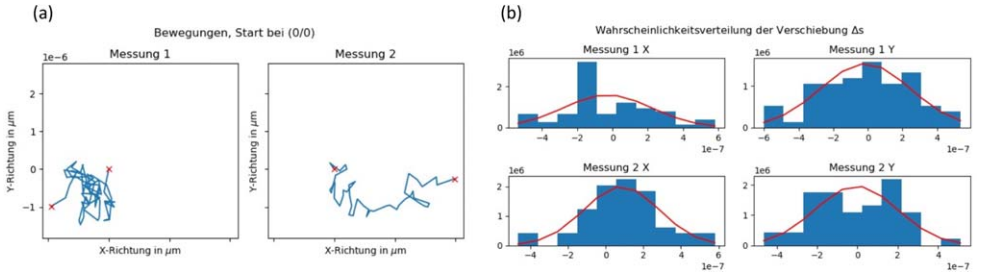

Figure 6. Extract from a student group's work on Brownian motion of a polystyrene particle in water. In (a) the experimental particle trajectories from two data sets obtained through video analysis of microscope images. In (b) students plot the relative frequencies of the particle's displacements divided by the bin size as a function of the particle's displacements.

figure 6 for an extract[4]) demonstrates solid computational skills in analysing the data. Similar capabilities were exhibited by other groups.

Conclusions

In this study, we successfully integrated computational aspects into an existing PLC, assuring that students could learn a programming language without feeling overwhelmed, while keeping the course structure largely unchanged.

To include Python data analysis in the laboratory course, we used the following guiding principles: (i) start defining a few clear learning goals, (ii) create specific activities to let students practice those aspects and foster the achievement of those outcomes, and (iii) assess students' achievement of those learning outcomes. Regarding (i), we concentrated on very specific learning outcomes we wanted students to be able to accomplish, which are related to (a) implementation of Python codes for data handling and analysis during the laboratory course experiments and (b) to communication aspects like the creation of correct, complete and readable graphs. This specificity allowed us to keep the already large number of existing learning goals in the introductory laboratory course while not overburdening students.

We adopted a gradual scaffolding in the curriculum in which students start working with Python during two lessons dedicated entirely to Python and Jupyter Notebooks and only then apply their knowledge to perform the data analysis in Python while doing experiments. We also adjusted our instruction accordingly to students' perspectives towards computation and previous experiences, using pre-post attitudinal surveys.

The introduction of carefully designed JNs, not only fostered the acquisition of programming skills that students can use in their future courses and careers, but also helped the development of expert-like skills in data analysis. The effectiveness of this approach was evidenced by assessments aligned with the laboratory computational learning goals defined in our design. Instead of depending on ready-made data analysis tools as we did in the past, which often hinder the finer details, students are now adapting and implementing data analysis techniques in Python from scratch, tailored to the specific requirements of their experiments. The intervention was well-received, with students maintaining positive attitudes

[4] As can be seen from the original excerpt, some of the graphs lack either axis description or scale. This behaviour, as is well known, is commonly found in PLC, and the use of Jupyter Notebooks has not impacted this habit.

into the following semester. Even with diverse backgrounds in programming, almost all students recognized in the pre- and post- questionnaires the importance of learning programming for data analysis and positively valued the early integration into the curriculum. We consider this aspect as very positive since the students' epistemology has an important role in learning [30]. In fact, when introducing research-based changes or innovations in a course students' resistance to change can be a challenge [33]. We believe that the positive attitude we found among our students helped to incentivize them to acquire computational skills. Moreover, we found that students continued to meet other existing course goals effectively [18], as measured with the E-CLASS instrument [19, 34, 35].

However, it is important to acknowledge some limitations in our study. The primary limitation being that our analysis of students' computational skills focused on group work, and therefore might not provide a comprehensive understanding of individual student outcomes, but only on the class as a whole. Additionally, given that the number of students considered in this study is small ($N < 50$), the statistical analysis is only descriptive and not inferential. As such, the conclusions may not apply in other contexts. In fact, our study focused on a specific class of first-year physics students, some of whom had a background in programming, which has become more common in recent years in German schools. In another country, the proportion of students with such a background might vary, although initiatives introducing programming in high school have become widespread globally.

For the future, we plan to monitor students further by collecting more data and analyse the long-term effects and benefits of our intervention. We are also aware that there is a need to reinforce the computational skills acquired in our introduction by providing further opportunities for practice in subsequent laboratory courses. In circumstances where several faculty instructors are involved, collaboration is required.

We also aim to introduce formative assessment moments tailored for individual students. The next implementation, under consideration, will be proposed focusing possibly on a single environment to facilitate the process. In conclusion, our findings indicate that it is both feasible and beneficial to integrate Python-based data analysis in a PLC improving students' computational skills without significantly altering the course structure.

Acknowledgments

We gratefully thank A Faber and R Reimann for helping with the English–German translations of the Jupyter Notebooks. MA thanks R Bausinger for introducing her to the use of Jupyter Notebooks and ET thanks Giovanni Organtini for introducing him to the use of Pandas.

Data availability statement

The data cannot be made publicly available upon publication because they contain sensitive personal information. The data that support the findings of this study are available upon reasonable request from the authors.

Ethical statement

The authors declare that an informed consent for participation and publication was obtained from all participants in this study. Consent covered the use of questionnaire responses and notebook analyses. The study was conducted according to the institution data protection rules.

ORCID iDs

Eugenio Tufino https://orcid.org/0000-0002-6784-3761
Stefano Oss https://orcid.org/0000-0001-9427-2151
Micol Alemani https://orcid.org/0000-0003-2004-8565

References

[1] World Economic Forum, (https://weforum.org/docs/WEF_Future_of_Jobs_2020.pdf) 2020 (accessed 31 May 2024)
[2] Behringer E, Burciaga J, Dietz D, Gavrin A, Kozminski J, Migenes V and Schroeder D 2016 AAPT recommendations for computational physics in the undergraduate physics curriculum, (https://aapt.org/resources/upload/aapt_uctf_compphysreport_final_b.pdf) (accessed 31 May 2024)
[3] Phys21: preparing physics students for 21st-century careers, (https://compadre.org/JTUPP/docs/J-Tupp_Report.pdf) 2016 (accessed 31 May 2024)
[4] Effective practice for computation, (https://ep3guide.org/guide/computational-skills) 2021 (accessed on May 31 2024)
[5] Caballero M D *et al* 2019 PICUP: a community of teachers integrating computation into undergraduate physics courses *Phys. Teach.* **57** 397–9
[6] Chabay R and Sherwood B 2008 Computational physics in the introductory calculus-based course *Am. J. Phys.* **76** 307–13
[7] Caballero M D and Pollock S J 2014 A model for incorporating computation without changing the course: an example from middle-division classical mechanics *Am. J. Phys.* **82** 231–7
[8] Odden T O B and Malthe-Sørenssen A 2021 Using computational essays to scaffold professional physics practice *Eur. J. Phys.* **42** 015701
[9] Burke C J and Atherton T J 2017 Developing a project-based computational physics course grounded in expert practice *Am. J. Phys.* **85** 301–10
[10] Sachmpazidi D, Bautista M, Chajecki Z, Mendoza C and Henderson C 2021 Integrating numerical modeling into an introductory physics laboratory *Am. J. Phys.* **89** 713–20
[11] Serbanescu R M, Kushner P J and Stanley S 2011 Putting computation on a par with experiments and theory in the undergraduate physics curriculum *Am. J. Phys.* **79** 919–24
[12] Beichner R, Chabay R and Sherwood B 2010 Labs for the matter & interactions curriculum *Am. J. Phys.* **78** 456–60
[13] Scherer D, Dubois P and Sherwood B 2000 VPython: 3D interactive scientific graphics for students *Comp. Sci. Eng.* **2** 56–62
[14] Atherton T J 2023 Resource letter CP-3: computational physics *Am. J. Phys.* **91** 7–27
[15] Leary A, Irving P W and Caballero M D 2018 The difficulties associated with integrating computation into undergraduate physics *Proc. of PER Conf. (Washington, DC)* (https://doi.org/10.1119/perc.2018.pr.Leary)
[16] American Association of Physics Teachers 2015 AAPT recommendations for the undergraduate physics laboratory curriculum, (https://aapt.org/resources/upload/labguidlinesdocument_ebendorsed_nov10.pdf) (accessed 31 May 2024)
[17] Hamerski P C, McPadden D, Caballero M D and Irving P W 2022 Students' perspectives on computational challenges in physics class *Phys. Rev. Phys. Educ. Res.* **18** 020109
[18] Alemani M 2023 The redesign of an introductory physics laboratory course *Il Nuovo Cimento*C **46** 181
[19] Teichmann E, Lewandowski H J and Alemani M 2022 Investigating students' views of experimental physics in German laboratory classes *Phys. Rev. Phys. Educ. Res.* **18** 010135
[20] Stanley J T and Lewandowski H J 2018 Recommendations for the use of notebooks in upper-division physics lab courses *Am. J. Phys.* **86** 45–53
[21] Alemani M 2023 Laborbücher: eine authentische form der wissenschaftlichen dokumentation mit didaktischem potenzial *Unterricht Phys.* **195** + **196** 14–9
[22] (2022) *SciDavis* (http://scidavis.sourceforge.net/)
[23] Weller D P, Bott T E, Caballero M D and Irving P W 2022 Development and illustration of a framework for computational thinking practices in introductory physics *Phys. Rev. Phys. Educ. Res.* **18** 020106

[24] Bandura A and (National Inst of Mental Health) 1986 *Social Foundations of thought and Action: A Social Cognitive Theory* (Prentice-Hall, Inc)
[25] (2022) *Jupyter* (https://jupyter.org)
[26] (2023) *Github Repository* (https://github.com/etufino/Jupyter-Notebooks-in-Lab-Course-Uni-Potsdam)
[27] (2022) *Colaboratory* (https://colab.research.google.com)
[28] (2022) *Anaconda Environment* (https://anaconda.org)
[29] Braun V and Clarke V 2006 Using thematic analysis in psychology *Qual. Res. Psychol.* **3** 77
[30] Lising L and Elby A 2004 The impact of epistemology on learning: a case study from introductory physics *Am. J. Phys.* **73** 372–82
[31] Smith E M and Holmes N G 2021 Best practice for instructional labs *Nat. Phys.* **17** 662–663
[32] Etkina E, Karelina A and Ruibal-Villasenor M 2008 How long does it take? A study of student acquisition of scientific abilities *Phys. Rev. Spec. Top. Phys. Educ. Res.* **4** 020108
[33] Deslauriers L, McCarty L S, Miller K, Callaghan K and Kestin G 2019 Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom *Proc. Natl. Acad. Sci. USA* **116** 19251–7
[34] Zwickl B M, Hirokawa T, Finkelstein N and Lewandowski H J 2014 Epistemology and expectations survey about experimental physics: development and initial results Phys *Rev. ST Phys. Educ. Res.* **10** 010120
[35] Wilcox B R and Lewandowski H J 2018 A summary of research-based assessment of students' beliefs about the nature of experimental physics *Am. J. Phys.* **86** 212