Journal of Physics:
Conference Series

PURPOSE-LED PUBLISHING™

PAPER • OPEN ACCESS

Using Jupyter Notebooks to foster computational skills and professional practice in an introductory physics lab course

To cite this article: Eugenio Tufino *et al* 2025 *J. Phys.: Conf. Ser.* **2950** 012022

View the article online for updates and enhancements.

You may also like

- Ultrafast ferroelectric field-effect transistors with low operating voltage based on van der Waals heterostructures
Dehuan Meng, Shuai Liu, Weiping Wang et al.
- Effect of Low-Temperature Tempering Temperature on the Mechanical Properties of NM500 Wear-Resistant Steel
Zheng Yihao, Ren Shuai, Zhang Xinyue et al.
- Simulation study on the growth of silicon carbide crystals using iron as a solvent
Haonan Li, Hongjun Wu, Xiuhua Chen et al.

This content was downloaded from IP address 213.55.247.6 on 23/02/2025 at 21:47

Journal of Physics: Conference Series **2950** (2025) 012022 doi:10.1088/1742-6596/2950/1/012022

Using Jupyter Notebooks to foster computational skills and professional practice in an introductory physics lab course

Eugenio Tufino[0000-0002-6784-3761, 1], **Stefano Oss**[0000-0001-9427-2151, 1] **and Micol Alemani**[0000-0003-2004-8565, 2]

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

eugenio.tufino@unitn.it

Abstract. In this paper, we detail the integration of Python data analysis into a first-year physics laboratory course, a task accomplished without significant alterations to the existing course structure. We introduced tailored laboratory computational learning goals and designed activities to address them. We emphasise the development and application of Jupyter Notebooks, tailored with exercises and physics application examples, to facilitate students' mastery of data analysis programming within the laboratory setting. These Notebooks serve as a crucial tool in guiding students through the core principles of data handling and analysis in Python, while working on simple experimental tasks. The results of the evaluation of this intervention offer insights into the advantages and challenges associated with early integration of computational skills in laboratory courses, providing valuable information for educators in the field of physics education. This study demonstrates a practical and effective way of embedding computational skills into the physics curriculum, and contributes to the ongoing efforts of the physics education research community.

1. Introduction

In the context of today's technological advancement, computational skills in physics are essential. The increased data generation in experimental sciences necessitates robust data analysis tools. This urgency is echoed in the physics education community's efforts to integrate computation into curricula and by recommendations from the American Association of Physics Teachers (AAPT) [1] and the Partnership for Integration of Computation into Undergraduate Physics (PICUP) [2].

Following these recommendations a variety of approaches to incorporate computation have been developed, ranging from adapting existing physics lectures to include computational elements [3, 4, 5], to the creation of project-based computational courses [6] and the integration of computational tools into laboratory settings. For instance, in high-school laboratory settings Wolfram Mathematica was used to introduce the use of simulations [15]. In introductory first-year laboratory courses, tools like Microsoft Excel [7] and VPython [8] modules have been utilised, while more advanced settings often employ programming languages such as Python [8]. For an extensive and current (2023) overview of the role of computation in physics education, refer to the Resource Letter mentioned in [9].

Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

This paper examines the introduction of Python for data analysis in a first-semester laboratory course. Python was chosen for its widespread use, market demand, flexibility, and comprehensive online documentation. Our goal aims to make programming more accessible to all physics major students from the outset of their university education, lowering the entry barrier to programming and fostering active student engagement. The challenges of embedding computational tools into already intensive first-semester laboratory courses, which focus on statistical analysis, measurement uncertainties, and experimental documentation, are addressed.

In this conference proceeding, the teaching approach, the definition of computational learning goals and the curriculum structure are presented and discussed. We showcase examples of developed Jupyter Notebooks (JNs) [10], illustrating the pedagogy implemented. These examples highlight the integration of computational tools into the curriculum and the methods used to engage students. While we explain the teaching approach and curriculum development in this paper, a comprehensive analysis of the assessment outcomes is available in reference [11], offering an in-depth evaluation of the effectiveness of these methods in fostering computational skills.

2. Our approach for integrating computational skill in the physics laboratory course

The design and implementation of our intervention involved three main steps. Initially, we defined specific laboratory computational learning goals, focusing on introductory Python data analysis skills. We then developed curricular materials to support these goals and facilitate student learning. The final step involved evaluating the achievement of these goals through analysis of student work and pre-post questionnaires.

We implemented this intervention in 2022-2023 in the University of Potsdam's first-semester laboratory course for physics majors.

This course is part of a progressive series of four modules during the first four semesters. The complete sequence was redesigned from 2016 to focus on developing experimental skills and encouraging expert-like thinking. Details of the course redesign and its assessment are presented elsewhere [12, 13]. Note that while introducing computational aspects into the course, we retained the original laboratory goals, slightly reducing in-class exercises and replacing the use of a scientific data analysis software SciDavis [14] with Python programming for data analysis.

The laboratory computational learning goals for Python were informed by the AAPT recommendations and relevant literature on computational thinking [1, 4, 5, 16], focusing on practical data analysis skills. Our goals were categorised into two areas, that we refer to as 'Implement' and 'Communicate'. In the first, we aim to enable students to write Python codes for problem-solving and for manipulating, analysing, and visualising data sets. In the second, we want students to learn how to effectively communicate their work using Jupyter Notebooks, write clear and well-commented Python code and create clear, informative, and comprehensive graphs using Python.

Our curriculum design for integrating computation involved using JNs for Python programming, which were applied by students also in experimental activities and tested in subsequent semesters. We created introductory and advanced JNs for both collaborative and independent learning. The material, accessible on GitHub [17], was tailored to be approachable even for students without prior coding experience.

We installed the Anaconda distribution for Python on our lab PCs for instruction. We did not propose Google Colab cloud-based platform (which would offer advantages like no installation requirement and collaboration features), due to data privacy considerations. However, we discussed this environment with students as well.

The assessment of the achievement of the defined computational learning goals involved both formative and summative methods, and students' attitudes towards computation were surveyed.

Table 1. Overview of the in class activities and data collection tool used to introduce Python data analysis in the course and to assess students skills acquisition.

Timeline	Data Collection Tool	Description of the Tool	Dimension Evaluated
Beginning of first semester	Pre-Survey	Pre-course survey completed individually by students.	Attitudinal and Emotional
Session 1 and Session 2	Jupyter Notebooks	Students work in groups, completing and submitting JNs (1-4) with their solutions to exercises.	Computational learning goals
First and second laboratory activities	Laboratory notebook	Students in groups submit either a JN or a lab notebook containing the data analysis of the activity.	Computational learning goals and lab learning goals
After laboratory activities	Post-Survey	Post-course survey completed individually by students.	Attitudinal and Emotional
Beginning of second semester exercises	Jupyter Notebooks	Students in groups submit either a JN or a lab notebook containing the data analysis of the activity.	Computational learning goals and lab learning goals

Importantly, this evaluation was distinct from the assessment of the existing laboratory learning goals, which were evaluated separately as in the previous years.

We considered emotional and attitudinal factors, such as students' confidence and self-efficacy [18], as well as their sense of authenticity in using Python as practising physicists do. Our primary aim was to lower the barrier to programming, cultivating a belief among all students, regardless of their prior experience, that they could actively engage in Python data analysis. Furthermore, we encouraged an authentic attitude of reading and understanding the codes written by others and building upon them, aligning with the practices of professional physicists. Our assessment encompassed surveys and ongoing monitoring during laboratory work. For instance, students were encouraged to perform data analysis directly while doing experiments in the laboratory sessions. Additionally, we dedicated effort to nurturing a positive attitude and self confidence among students in their use of Python for data analysis, ultimately enhancing their computational skills.

Table 1 provides an overview of the activities and assessment tools employed in this study. It includes a description of each tool and specifies the dimension being evaluated. The data were collected from 48 students who voluntarily participated in the study. The pre- and post-course surveys were completed individually, with 42 matched responses obtained for analysis. Other tasks, such as Jupyter Notebook activities and lab notebook submissions, were conducted in group settings throughout the course.

3. Description of the Jupyter Notebooks developed

In this section, we provide a detailed description of the JNs that were prepared by us for first year introductory laboratory course physics students. Our approach is to teach programming concepts within the context of physics, using examples related to calculating physical quantities or simulating simple physical systems, thereby enabling students to directly relate programming to their field of study. Students learn to import, manipulate, and analyse data using Python. The libraries used were math, numpy (and numpy random), matplotlib, scipy (and scipy optimize) and pandas. Some additional libraries for handling data in Google Colab were used in the additional material for students. All exercises were conducted in groups in class, apart from those described in sections 3.2 and 3.3 that were completed asynchronously and independently by students to allow to explore additional Python's capabilities. Each lab session lasted for three hours. Students, working normally in groups of three, could use Python programming either on their PCs or their own device.

They implement using Python techniques like linear regression, error analysis, and graphical data representation, previously introduced and reinforced throughout the course. Students practise these concepts and techniques in real-world scenarios. The curriculum is divided into three modules, each focusing on a specific aspect of Python data analysis:

3.1. Introduction to Python (four JNs):

This module provides a comprehensive introduction to Python programming, covering essential concepts such as installation, basic calculations, text formatting, LaTeX usage, and creating simple plots. Furthermore, the implementation of the least squares method for linear regression and the calculation of statistics parameters like the coefficient of correlation and the chi-squared are treated (manually calculated by formulas). After this introduction, done in two (3h each) in class sessions, students engage in two in-person lab sessions, each lasting three hours, to gain hands-on experience with these data analysis concepts and techniques.

3.2. Application Examples (five JNs):

This module applies Python data analysis to real-world physics phenomena. Students explore exercises that involve calculating the moving average applied to sunspot time-series, plotting and analysing data of light source illuminance vs. 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 a website.

3.3. Additional Python Techniques (two JNs):

This module focuses on mastering data import and manipulation techniques across various platforms. Students learn how to effectively import and manipulate data files using Jupyter Lab-Anaconda, Google Colab, and online sources. These techniques are acquired through asynchronous independent work, allowing students to further extend their abilities and apply them to future projects.

Through this structured curriculum, students can gain a solid foundation in Python data analysis, enabling them to effectively analyse and interpret real-world data in their physics studies and beyond. Note that our intervention did not significantly alter the course structure. The following Section describes sample activities with Jupyter Notebooks used in the course.

4. Examples of Jupyter Notebooks

4.1. Lesson 2's second JN: exploring Pandas and the least squares method

In the second lesson dedicated to Python (Notebook 4), we introduced the use of Pandas, a Python library widely used in professional data analysis. This session built on the students'

Exercise 1:

In an experiment, a stone is thrown vertically upwards with initial velocity v. A student makes seven throws to check if the relationship between v^2 and the stone height h is of the form $v^2 = 2gh$.

The student's data in a Pandas data frame is shown in the following cell.

Assign h and v^2 to the variables x and y and then plot v^2 against h including vertical bars associated with the uncertainties of y.

```
[ ] #Data

    # we define here the data with a dictionary and a list
    stone = {'h (m)': [0.4,0.8,1.4,2.0,2.6,3.4,3.8], 'v^2 (m/s)^2': [7,17,25,38,45,62,72],
             'Uncertainties on v^2 (m/s)^2':[3,3,3,4,5,5,6]} # we define here the data, by using dictionary and list.
    stone_df=pd.DataFrame(data=stone)
    stone_df.head()
```

	h (m)	v^2 (m/s)^2	Unsicherheit von v^2 (m/s)^2
0	0.4	7	3
1	0.8	17	3
2	1.4	25	3
3	2.0	38	4
4	2.6	45	5

Figure 1. Snapshot of a JN with which students can practice creating a Pandas dataframe.

existing knowledge of graphing data and of the least squares method, marking a transition from traditional paper and pencil method to a computational approach. The notebook starts with a comprehensive introduction to Pandas, including practical guidance on how to import data into Pandas DataFrames and highlighting its crucial role in physics data analysis. A central part of the notebook is an exercise focused on the motion of a vertically thrown stone (see figure 2). In this exercise, students are asked to analyse the stone's trajectory, aiming to test the applicability of the relationship between the stone velocity v and its height h as $v^2 = 2gh$, where g is the acceleration due to gravity.

This exercise involves importing data into a DataFrame, performing a linearisation of the data and applying the least squares method for linear regression and finally the calculation of the reduced chi-squared. The screenshots included below (figure 2) show this progression. Students are first introduced to basic data handling using Pandas, progressing to more complex concepts like residuals analysis, and eventually to weighted least squares and reduced chi-square analysis. This approach not only makes the exercise scientifically significant but also allows students to see the practical application of these computational methods in a real-world physics context.

4.2. Example from JN applications series: CO_2 plot

In this asynchronous application example (shown in figure 3), we engage students in plotting data of atmospheric CO_2 concentration over time, a topic of profound scientific and environmental significance. An important initial step in the task is for students to examine the CSV data file from the scientific website NOAA Global Monitoring Laboratory. This examination is crucial as it helps students understand the file's structure and determine the starting point of the data. They learn to use the `skiprows` function in Python to correctly import the data, bypassing non-essential header information. This introduces them to a key aspect of data handling: preprocessing and cleaning data for analysis. Following this, students graphically represent the CO_2 data over time. This step not only illustrates the application of Python in data visualisation but also brings attention to the ongoing changes in atmospheric CO_2 levels.

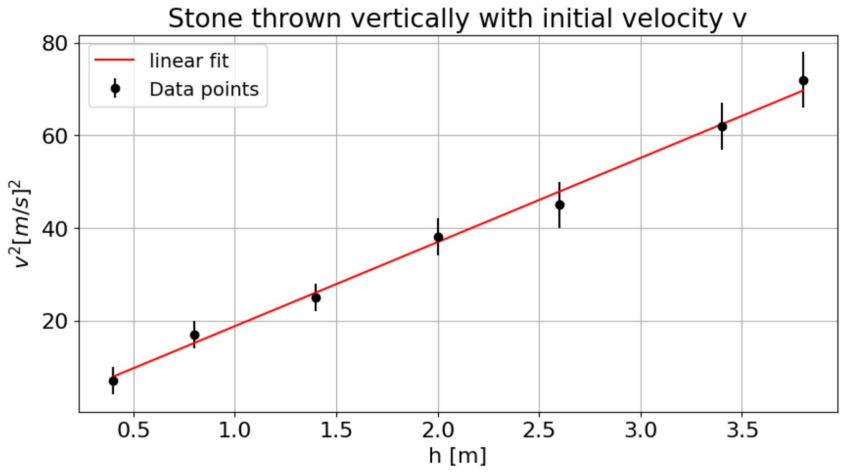
Exercise 4:

Find the **weighted** best fit line for the stone experiment above and plot your results. Find the slope and discuss if the results are consistent with the value of $g = 9.81 m/s^2$. Calculate the reduced chi squared as well. Notice that the data are stored in the stone_df dataframe.

```
[33] #Solution part 1
     params, params_covariance = curve_fit(linear_fit, x_stone, y_stone,sigma=error_y_stone,absolute_sigma=True)
     print(params) #Optimal values for the parameters so that the sum of the squared residuals is minimized.
     slope = params[0] # slope b
     intercept = params[1]   # intercept
     u_intercept = np.sqrt(params_covariance[1][1]) # see above
     u_slope = np.sqrt(params_covariance[0][0])   #see above
     print(params_covariance)

     [18.17440549  0.58972722]
     [[ 1.83372795 -2.76525743]
      [-2.76525743  6.1556603 ]]
```

```
[34] ##Solution part 2
     #graph and fit
     plt.errorbar(x_stone,y_stone,yerr=error_y_stone,color='black',fmt='o',label='Data points')
     plt.plot(x_stone, slope*x_stone+intercept,'r',label='linear fit')
     plt.title("Stone thrown vertically with initial velocity v")
     plt.legend(loc='upper left')  # changed the legend location with keyword loc
     plt.xlabel('h [m]')
     plt.ylabel('$v^2 [m/s]^2$')
     plt.grid(True)
     plt.show()
     print("The best fit line has the following parameters: ")
     print("==================y=bx+a=======================\n")
     print("The slope of the linear_fit function is:{:5.2f} m/s^2 with uncertainty {:5.2f} m/s^2".format(slope,u_slope))
     print("The intercept of the linear_fit function is:{:5.2f} (m/s)^2  with uncertainty {:5.2f} (m/s)^2".format(intercept, u_intercept))
     print("====================Chi square==================\n")
     #yerr=error_y_fall
     print("The reduced chi square is equal to: {:5.2f}".format(reduced_chi_square(x_stone,y_stone,error_y_stone,linear_fit, *params)))
```




```
The best fit line has the following parameters:
==================y=bx+a=======================

The slope of the linear_fit function is:18.17 m/s^2 with uncertainty  1.35 m/s^2
The intercept of the linear_fit function is: 0.59 (m/s)^2  with uncertainty  2.48 (m/s)^2
====================Chi square==================

The reduced chi square is equal to:  0.23
```

Figure 2. Snapshot of a JN with which students can practice plotting data and perform a linear regression and calculate the chi-squared.

Plotting CO_2 data

In this Notebook we want to practice plotting data. To start, we will plot the monthly mean of carbon dioxide measured at Mauna Loa Observatory, Hawaii measured from the 1958. The data are from the webpage under this link: https://gml.noaa.gov/ccgg/trends/data.html
There you can find detailed information about how this data was taken and find additional interesting data to plot.

```
[ ] # import here all the packages we need
    import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
```

```
[ ] plt.rcParams['figure.figsize'] = (10,5) # size of figure
    plt.rcParams['font.size'] = 16    # fonsize of the text
    plt.rcParams['legend.fontsize'] = 14  # set fontsize of the legend
```

```
[ ] dataCO2 = pd.read_csv('https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_mm_mlo.csv',skiprows = 52)
    # with skiprows we don't consider the first rows with textual comment
```

```
[ ] dataCO2.head() #we take a look to the data
```

	year	month	decimal date	average	deseasonalized	ndays	sdev	unc
0	1958	3	1958.2027	315.70	314.43	-1	-9.99	-0.99
1	1958	4	1958.2877	317.45	315.16	-1	-9.99	-0.99
2	1958	5	1958.3699	317.51	314.71	-1	-9.99	-0.99
3	1958	6	1958.4548	317.24	315.14	-1	-9.99	-0.99
4	1958	7	1958.5370	315.86	315.18	-1	-9.99	-0.99

Be reading on the webpage of the 'Global Monitoring Laboratory' we understand that the column 'average' represents the mole fraction of carbon dioxide with units 'part per million' or 'ppm'. The monthly mean values are centered on the middle of each month. The column 'deseasonalized' is the same as the column 'average', but after correction for the average seasonal cycle.

We decide to plot here in the y-axis the data in 'deseasonalized' and on the x-axis the year the data was taken.

```
[ ] x_CO2 = dataCO2['year'] #assign to x the values in the column 'year'
    y_CO2 = dataCO2['deseasonalized'] #assign to y the values in the column 'average'
```

```
    # plot by using Pandas feature
    dataCO2.plot('year' ,'deseasonalized', color='red', label='Data taken at Scrips Institution of Oceanography and\nNOAA Global Monitoring Laboratory')

    # plot by using plt and the variable x and y assigned above
    #plt.plot(x_CO2 ,y_CO2, 'r', label='Data taken at Scrips Institution of Oceanography and\nNOAA Global Monitoring Laboratory')
    plt.grid(True)
    plt.legend()
    plt.xlabel('Year')
    plt.ylabel('Mole fraction of carbon dioxide (ppm)')
    plt.title("Atmospheric CO$_{2}$ at Mauna Loa Observatory")
    plt.show()
```

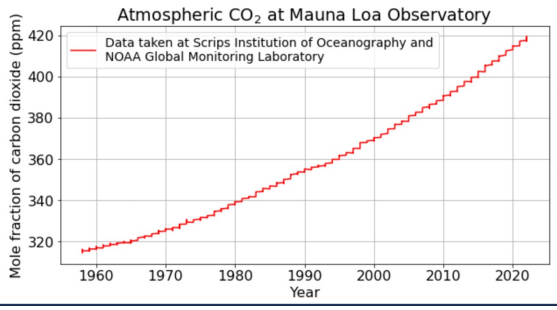


Figure 3. Snapshot of a JN with which students can practice plotting data.

5. Assessment of students' computational skills

This section examines the assessment of students' proficiency in using Python and Jupyter Notebooks (JNs) for data analysis in an introductory laboratory setting. The details of the assessment are described in a separate publication [11]. We analysed students' work using a rubric with two categories: 'Implement' and 'Communicate' which are aligned to our defined learning goals. At the following link [17] the rubric used to evaluate the laboratory computational skills (see Figure 4) is available. Each category contains a set of sub-skills (I1 and I2 for the two Implement related skills and C1, C2, C3 for the three Communicate related skills). In the sessions S1, S2, S3, all sub-skills were considered. For the laboratory experiments we evaluated only the sub-skills I1, I2, C1, C3.

The rubric has 4 levels of ability, 'missing', 'inadequate', 'needs some improvement' and 'near mastery'. In figure 4 we present the results of the assessment, showing the percentage of students that reached each level of ability in the different in-class activities. In the following we discuss them separately.

5.1. First sessions: Introduction to Python (sessions S1 and S2)

The high scores prevalent in these sessions indicate an effective grasp of Python programming essentials early in the course, particularly in skills related to 'Implement'.

5.2. Laboratory Experiments in the First Semester (LAB1 and LAB2)

Following the introductory Python sessions, students participated in two laboratory experiments, the first involving stretchable objects to explore Hooke's law [19] and the second a study on pendulum. Our analysis of students' laboratory notebooks revealed an upsurge in Python and JN utilisation, as illustrated in figure 4 (LAB1 and LAB2). The assessments indicate proficient use of Python in data analysis, with commendable scores in both 'Implement' and 'Communicate' categories. Notably, a progression in skill mastery is observed from LAB1 to LAB2. Python and JN, while optional, were increasingly adopted by students, evident from 83% usage in LAB1 to complete adoption in LAB2.

5.3. Continued Proficiency in the Second Semester (S3)

The beginning of the second semester involved revisiting Python skills through two review exercises, the outcomes of which are depicted in figure 4 as S3. These exercises, assigned after a three-month interruption, served as a test for the retention of computational skills. The analysis shows that about 83% of the groups scored 2 or 3 in both 'Implement' and 'Communicate,' showing that students' mastery of the skills imparted in the previous semester lasted over the break. However, the presence of some groups scoring 1 (inadequate) underscores the variability in skill retention.

We finally note here that in this proceedings we briefly mentioned the findings related to students' attitudes towards introducing computation in the physics laboratory course. A detailed analysis and discussion are provided in [11]. Here we just refer that the analysis of the questionnaires (see table 1) suggests that students generally hold very positive views about this integration and notably maintain this positive perspective throughout the duration of our intervention. This positive outlook is crucial, as it is well-known that students' epistemological beliefs about a subject can significantly influence their learning process [20]. This understanding could encourage other educators to incorporate computational components into their laboratory courses.

6. Conclusions

In our study, we embedded computational aspects into an existing physics laboratory course, enabling students to learn programming seamlessly within the familiar course structure. The

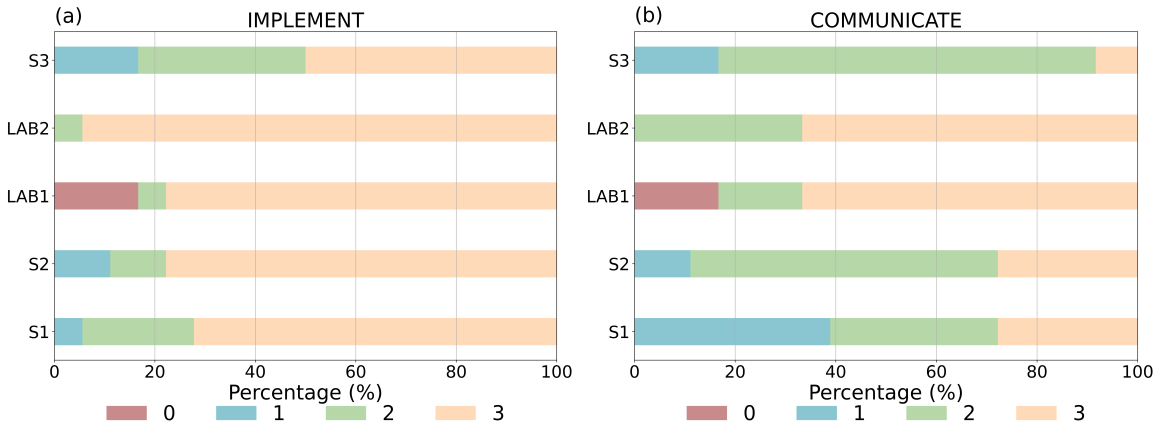


Figure 4. Percentage distribution of student scores in JNs across different course activities, subdivided into 'Implement' (a) and 'Communicate' (b). The Python introduction sessions (S1 and S2), the first semester laboratory experiments (LAB1 and LAB2), and the in-class exercises at the beginning of the second semester (S3) are shown distinctly. The scores range from 0 (missing), 1 (inadequate), 2 (needs some improvement), to 3 (near mastery).

introduction of Jupyter Notebooks played a crucial role, not only in equipping students with important programming skills for their academic and professional futures but also improving their expertise in data analysis.

This methodology proved effective, as shown by the assessments that were in line with the computational learning goals we set. A shift from using preset data analysis tools to applying Python for tailored data analysis was well-received by students, who kept using the learned skills in the following semester. This positive reception is significant in the context of introducing research-based changes, which often encounter student resistance [21].

Despite these successes, it's important to acknowledge the limitations of our study. The focus on group-based assessments may not fully reflect individual learning outcomes, and the small sample size limits the generalizability of our results. Our study's context, involving first-year physics students, some with prior programming experience, might also differ in other educational settings.

Among our future directions, we plan to continue monitoring students to assess the long-term impacts and benefits of our intervention, and to reinforce the computational skills acquired by providing further practice opportunities in subsequent laboratory courses. Additionally, we recognize the need to adapt our teaching strategies to the evolving educational landscape, particularly considering the advent of AI programming support, assisted by Large Language Models (LLMs) like ChatGPT. These AI tools are rapidly transforming how we approach teaching and learning. A balanced approach is essential. It's not about wholeheartedly adopting these tools, but rather exploring their potential to augment learning. This exploration should focus on enhancing educational experiences while ensuring students acquire essential skills.

Our experience indicates that integrating Python into physics laboratory courses effectively improves computational skills while maintaining the course structure.

References

[1] 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

(American Association of Physics Teachers) URL https://www.aapt.org/resources/upload/aapt_uctf_compphysreport_final_b.pdf
[2] Caballero M D, Chonacky M N, Engelhardt L, Hilborn R C, del Puerto M L and Roos K R 2019 PICUP: A community of teachers integrating computation into undergraduate physics courses *Phys Teach* **57** 397–9
[3] Chabay R and Sherwood B 2008 Computational physics in the introductory calculus-based course *Am J Phys* **76**(4) 307–13
[4] 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**(3) 231–7
[5] Odden T O B and Malthe-Sørenssen A 2021 Using computational essays to scaffold professional physics practice *Eur J Phys* **42**(1) 015701
[6] Burke C J and Atherton T J 2017 Developing a project-based computational physics course grounded in expert practice *Am J Phys* **85**(4) 301–10
[7] Sachmpazidi D, Bautista M, Chajecki Z, Mendoza C and Henderson C 2021 Integrating numerical modeling into an introductory physics laboratory *Am J Phys* **89**(8) 713–20
[8] 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**(9) 919–24
[9] Atherton T J 2023 Resource Letter CP-3: Computational physics *Am J Phys* **91**(1) 7–27
[10] Jupyter Notebook URL https://jupyter.org/
[11] Tufino E, Oss S and Alemani M 2023 Integrating Python data analysis in an existing introductory laboratory course *Eur J Phys* **45**(4) 045707
[12] Alemani M 2023 The redesign of an introductory physics laboratory course *Il Nuovo Cimento C* **46**(5) 5
[13] 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**(1) 010127
[14] SciDAVis website URL https://scidavis.sourceforge.net/
[15] Samsonau S V 2018 Computer simulations combined with experiments for a calculus-based physics laboratory course *Phys Educ* **53**(5) 053002
[16] 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**(1) 010135
[17] GitHub Repository for Jupyter Notebooks in Lab Course at Uni Potsdam URL https://github.com/etufino/Jupyter-Notebooks-in-Lab-Course-Uni-Potsdam
[18] Bandura A 1986 *Social Foundations of Thought and Action: A Social Cognitive Theory* (Englewood Cliffs, NJ: Prentice-Hall)
[19] Smith E M and Holmes N G 2021 Best practice for instructional labs *Nature Phys* **17**(6) 662–3
[20] Lising L and Elby A 2005 The impact of epistemology on learning: A case study from introductory physics *Am J Phys* **73**(4) 372–82
[21] 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**(39) 19251–7