

ENHANCING UNIVERSITY TEACHING AND LEARNING THROUGH A PYTHON TRAINING COURSE FOR EDUCATORS AND STUDENTS

**Manuel Gil-Martín, Sergio Esteban-Romero, Iván Martín-Fernández,
Juan Manuel Montero-Martínez, Rubén San-Segundo, & Fernando Fernández-Martínez**
*Grupo de Tecnología del Habla y Aprendizaje Automático (THAU Group),
Grupo de Innovación Docente en Ingeniería y Sistemas Electrónicos (GRIDS),
Departamento de Ingeniería Electrónica, Information Processing and Telecommunications Center,
E.T.S.I. de Telecomunicación,
Universidad Politécnica de Madrid (Spain)*

Abstract

Over the past few years, Python has emerged as one of the most versatile and widely used programming languages, ranging from data analysis to signal processing and artificial intelligence. Although gaining prominence in academia and industry, even today many university instructors are unaware about Python and lack official structured resources to learn such skills. To meet this requirement, we have designed a training course called "Introduction to the Python Programming Language" at the Institute of Educational Sciences of the Universidad Politécnica de Madrid. It is addressed both to lecturers and students and promotes digital literacy. The course offers elementary and advanced content, taking participants through simple constructs of programming, such as, variables, control structures, elementary data structures and data visualization using Matplotlib, to more advanced topics, such as, object-oriented programming, pandas-based data analysis, scientific computing with NumPy and SciPy, or symbolic computation with SymPy. An interesting feature of the course is use of interactive notebooks, completely executable in the Google Colab environment. This approach avoids installation limitations, provides flexibility between machines, and allows immediate experimentation, democratizing programming among instructors from different university faculties since only a web browser is required. The notebooks are designed as guided tutorials: aside from explaining the concepts, they include problem-solving exercises that prompt participants to experiment on their own and enhance their computational thinking skills. The course also incorporates educational innovation techniques such as problem-based learning, gamification, challenge-based learning, and active learning, which foster motivation, engagement, and deeper understanding through hands-on activities. Over several editions, the course has received growing interest, achieving full enrolment annually. Feedback from participants is consistently positive regarding the clarity, accessibility, and scope of the material, and the balance between theory and practical exercises. Participants particularly value the notebooks they can return to explanations while gradually moving towards increasingly challenging applications. The course success validates the relevance to offering specialized training in computer skills to the university community. The course aligns with generic training goals: it increases teaching quality by equipping teachers with tools to include computer programming in their lessons; enhances critical thinking and independence among students; and encourages lifelong learning within the academic community. The work presents the course design, implementation, and evaluation in the context of teacher training, curriculum development, and innovation in higher education. The results demonstrate how continuous professional development in programming can improve the teaching-learning experience and quality of education in a more digitalized society.

Keywords: *Higher education, teacher training, digital competencies, lifelong learning.*

1. Introduction

Python is a high-level, interpreted programming language useful thanks to its readability, simplicity, and versatility. Its clear syntax and rich ecosystem make it accessible to beginners while remaining powerful for advanced users (Chun, 2006). Indeed, Python has become instrumental in academia

and industry because it enables quick prototyping, reproducible research, and easy integration with scientific libraries (Otahanov, 2023).

Python supports a wide range of applications, from data analysis using NumPy and pandas to visualization using Matplotlib and seaborn (Paffenroth et al., 2013; Anderson, Preston, & Silva, 2002). It is also widely used in machine learning (Esteban-Romero et al., 2024; Martín-Fernández et al., 2024), signal and image processing, simulations, and interactive computing. This way, it is highly relevant for teaching and research.

In computer science and engineering programs, Python allows students to learn fundamental concepts without complex syntax, while in scientific degrees it supports experiments computation (Bădică et al., 2020). Concerning teachers, Python enables the creation of high-quality plots and small programs to show how algorithms or signal processing operations work (Anderson, Preston, & Silva, 2002; Müller, 2019). Interactive notebooks in environments like Google Colab remove installation barriers, encourage hands-on learning, and allow students to experiment in real time (Tufino, Oss, & Alemani, 2024), integrating computational thinking naturally into courses. The course is focused on providing tools for teachers and students of computer science and engineering programs to improve their teaching-learning experience.

2. Overview of the proposed course

The proposed course aims to enhance educators' and students' skills in the digital and computational field by offering a structured introduction to modern programming practices through the Python language. It is designed to support participants in developing both foundational and advanced competencies that can be applied to teaching, research, and professional activities across diverse academic fields from elementary aspects to advanced contents. This section provides a description of the course, including its objectives and content, and the evolution of the course during the academic years.

2.1. Learning goals and course content

This course offers an application-oriented view of the Python programming language. This section below offers a general overview of the objectives and contents of the course currently offered to educators and students at the Institute of Educational Sciences of the Universidad Politécnica de Madrid. Four 3-hour sessions are used for the course, scheduled two per semester during the academic year.

The general goals of the course are:

- Become familiar with the Python programming language.
- Acquire the basic and applied knowledge needed to continue learning Python independently (simple operations, control structures, functions).
- Explore general libraries such as NumPy and Matplotlib for array managing and visualization.
- Master advanced Python tools, especially data analysis and manipulation with pandas.
- Understand and apply object-oriented programming concepts, including inheritance and polymorphism.
- Explore specialized libraries such as SciPy and SymPy for scientific and symbolic computation.
- Introduce the use of databases in Python.

The course content is organized progressively, beginning with elementary aspects and advancing toward increasingly sophisticated techniques, including the following contents:

- Basic Python concepts: what Python is, its philosophy, and initial exercises using Google Colab.
- Fundamental operations: numerical operations, string manipulation, type conversion, lists, tuples, dictionaries, and slicing.
- Control structures: comparisons, conditionals, while and for loops, ranges, and common list operations.
- Functions and modules: code reuse, function syntax and arguments, documentation with docstrings, functions as objects, modules, and standard libraries.
- Core scientific libraries: use of NumPy for numerical computation and Matplotlib for data visualization.
- Data analysis and manipulation: introduction to pandas and its practical applications.
- Specialized libraries: scientific computing with SciPy and symbolic mathematics with SymPy.
- Object-Oriented Programming (OOP): classes, objects, attributes, and methods.
- Inheritance and polymorphism: extension of OOP principles for more advanced program design.
- Databases: first steps for connecting to, querying, and managing databases in Python.

This content provides the participants with a strong and flexible foundation for their application of Python effectively in academic, scientific, and professional fields, allowing them to take further advance into specialized areas of programming and data analysis.

2.2. Course evolution and redesign

This section analyzes how the Python course evolved over several academic years and how participants' feedback shaped its redesign. The course was originally organized (from 2021 to 2024) as two 4-hour intensive sessions in the first semester of the academic year. Based on the recurring feedback received, it was restructured in 2024–2025 into four separate modules, each lasting 3 hours, with the intention of improving pacing, structure, and learning outcomes. This transition from 8-hour intensive sessions to a four-module format totaling 12 hours and distributed over different semesters greatly improved the overall quality of the course. This extended duration not only allowed the inclusion of additional content but also provided participants with more opportunities to practice through in-class coding exercises at the end of each module. Participants were able to work through problems independently and raise questions more effectively through innovative techniques such as gamification or challenge-based learning. In addition, this redesign enabled the explanation of concepts at a slower and more comfortable pace, directly addressing one of the most recurrent issues in earlier editions. In parallel, a transition of computing environments was implemented. Initial editions relied on locally installed Jupyter Notebooks via Anaconda: participants had to configure the environment in advance, and this often generated setup issues that needed to be resolved during the first minutes of the course. The new format adopted Google Colab Notebooks. With Colab, participants only needed a Google account and ran all experiments on Google's servers, avoiding installation barriers and substantially reducing technical friction during the sessions. To document this evolution, the two following subsections include direct original comments from survey completed by the participants (kept exactly as they were written) after finishing the course.

2.2.1. Summary of strengths and weaknesses. This subsection summarizes the most frequent patterns found in the feedback. The weaknesses correspond to issues that appeared repeatedly across multiple editions during initial format, while the strengths reflect what participants consistently valued. Each point is accompanied by representative quotations expressed by the attendees.

- Weaknesses:
 - Insufficient duration:
 - “Too much content in too little time.”
 - “Very dense courses.”
 - “When it starts to get interesting, it ends.”
 - Pace too fast:
 - “Hard to follow if you don't have prior background.”
 - “The instructor is forced to rush because of the limited time.”
 - “Not enough time to absorb the concepts.”
 - Too little practice / insufficient exercises:
 - “There's no time to do exercises.”
 - “A practice-only session would be great.”
 - “I need more exercises between sessions.”
 - Sessions not spaced enough:
 - “Courses concentrated in 1–2 days don't allow anything to sink in.”
 - “It would be better to spread out the sessions.”
 - Dependence on the Anaconda installation environment:
 - “Initial environment setup delays starting the course.”
- Strengths:
 - Quality of materials:
 - “Fantastic documentation.”
 - “Very complete, very useful material.”
 - “Notebooks well prepared and progressive.”
 - The instructor / teaching:
 - “Excellent instructor.”
 - “Very clear, very didactic.”
 - “The instructor explains everything perfectly.”
 - Very interesting and motivating course:
 - “Makes you want to keep learning.”
 - “Very useful for my work.”
 - “Gives a good general overview of the language.”

- Good structure / progression:
 - “Very progressive material.”
 - “Easy to follow.”
 - “The course is well organized and practical.”
- Transition to Google Colab:
 - “The environment is simple and avoids configuration problems.”

2.2.2. Before–after comparison. This subsection shows Table 1, which compares the feedback from the initial 8-hour format (2021–2024) with that from the redesigned final four-module of 3 hours format (12 hours in total from 2024–2025). These comments are focused on three crucial aspects which were improved: the evaluation of the teaching, the structure of the material and the continuity of the course.

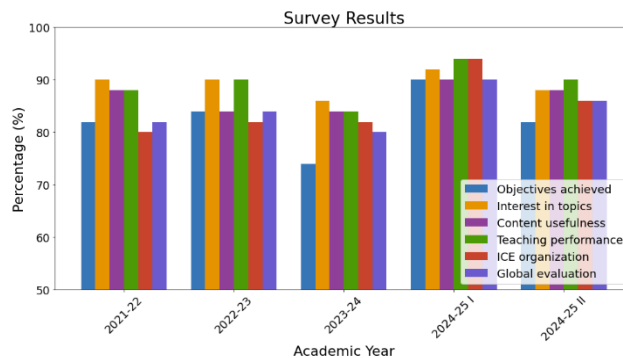
Table 1. Comments of the initial and final course format from the participants.

Aspect	Comments of the initial format	Comments of the final format
Evaluation of the teaching	“The pace is too fast.” “There aren’t enough pauses.”	“The professor explains very clearly.” “The course adapts very well to beginners.”
Clarity and structure of the material	“A roadmap is required.” “An introduction for beginners could be useful.”	“Very progressive material.” “Well organized.” “Material lets you continue learning on your own.”
Continuity of the course	“We need a follow-up course.” “What do I do after this?”	“I’d like to have a guaranteed spot in Python II.” “Great idea to split it into two modules.”

3. Discussion of survey results

The participants completed a survey after finishing the course, whose results are available in Figure 1. Some interesting trends in the course evaluation may be drawn from the survey results across past academic years. Overall, most items received consistently high scores. For instance, “Interest in topics” and “Teaching performance” normally scored over 85%, indicating sustained engagement and effective instruction. Notably, the first part of 2024–25 edition showed a noticeable improvement compared to the previous ones for all items, especially items such as “Objectives achieved” and “Content usefulness”, with scores around 90%, reflecting successful course enhancements. Moreover, the second part of 2024–25 kept strong performance, although slightly lower than the first, indicating that while improvements are sustained, there may still be room for fine-tuning the advance part of the course. “ICE organization” and “Global evaluation” items, which measures course coordination with the Institute of Educational Sciences and a general view of the course, show steady improvement over the years, peaking at 90-94% in 2024–25, suggesting that organizational changes, probably due to the transition from the 8-hour to the 12-hour course, had a positive impact on the student experience by reducing technical barriers.

Figure 1. Survey results during different academic years.



4. Conclusions

In conclusion, the “Introduction to the Python Programming Language” course effectively provided evidence for how structured and hands-on programming training could enhance digital literacy, computational thinking, and teaching quality across the university community. With iterative redesigns guided by participants’ feedback, most notably, splitting the original format of 8 hours into four 3-hour

modules and transitioning from locally installed Jupyter/Anaconda environments to Google Colab, both learning outcomes and satisfaction have substantially improved. The participants of the course consistently valued the clarity of instruction, the progressive structure of the materials, and the access to interactive notebooks, allowing self-directed experimentation and further learning beyond the sessions. Survey results confirm continuous engagement, relevance of the content, and high levels of satisfaction, highlighting the course's role in raising both educator and student competencies in programming. Overall, this initiative shows that targeted professional development in Python not only equips instructors with practical digital tools but also offers active learning and lifelong skill development across disciplines.

Acknowledgments

The work leading to these results was funded by the Spanish Ministry of Science and Innovation through the project TRUSTBOOST (PID2023-150584OB-C21) and by the European Union "NextGenerationEU/PRTR". Sergio Esteban-Romero's research was supported by the Spanish Ministry of Education (FPI grant PRE2022-105516). Iván Martín-Fernández's research was supported by the Universidad Politécnica de Madrid (Programa Propio I+D+i). The authors gratefully acknowledge Instituto de Ciencias de la Educación of Universidad Politécnica de Madrid for their valuable support in the organization and coordination of the course during different academic years.

References

- Anderson, E. W., Preston, G. A., & Silva, C. (2002). Using Python for Signal Processing and Visualization. *Computing in Science & Engineering*, 4(2), 62-71.
- Bădică, C., et al. (2020). On the Role of Python in Programming-related Courses for Computer Science and Engineering Academic Education. *Education and Information Technologies*, 25(6), 5431-5447.
- Chun, W. J. (2006). *Core Python Programming*. Upper Saddle River, NJ: Prentice Hall.
- Esteban-Romero, S., Martín-Fernández, I., Gil-Martín, M., Griol-Barres, D., Callejas-Carrión, Z., & Fernández-Martínez, F. (2024). LLM-driven Multimodal Fusion for Human Perception Analysis. *Proceedings of the 5th on Multimodal Sentiment Analysis Challenge and Workshop: Social Perception and Humor (MuSe'24)* (pp. 45-51). New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3689062.3689084>
- Martín-Fernández, I., Esteban-Romero, S., Bellver-Soler, J., Fernández-Martínez, F., & Gil-Martín, M. (2024). Larger encoders, smaller regressors: Exploring Label Dimensionality Reduction and Multimodal Large Language Models as Feature Extractors for Predicting Social Perception. *Proceedings of the 5th on Multimodal Sentiment Analysis Challenge and Workshop: Social Perception and Humor (MuSe'24)* (pp. 20-27). New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3689062.3689083>
- Müller, M. (2019). PCP Notebooks: A Preparation Course for Python with a Focus on Signal Processing. In A. B. Smith & C. D. Johnson (Eds.), *Advances in Python Education* (pp. 45-60). Berlin, Germany: Springer.
- Otahanov, N. A. (2023). Teaching the Python programming language in higher education institutions. *Computer Applications in Engineering Education*, 31(5), 1567-1576.
- Paffenroth, R., et al. (2013). Python in Data Science Research and Education. *Proceedings of SciPy*, 21-28. Retrieved November 18, 2025, from <https://proceedings.scipy.org/articles/Majora-7b98e3ed-019.pdf>
- Tufino, E., Oss, S., & Alemani, M. (2024). Using Jupyter notebooks to foster computational skills and professional practice in an introductory physics lab course. *arXiv preprint arXiv:2405.16675*. Retrieved November 18, 2025, from <https://arxiv.org/abs/2405.16675>