

CASE STUDY: Integration of Python programming in a civil engineering laboratory course

Prof. John S Popovics P.E., University of Illinois at Urbana - Champaign

John Popovics is a Professor in the Civil and Environmental Engineering Department at the University of Illinois at Urbana-Champaign. He earned his B.S. and M.S. in Civil Engineering from Drexel University and his Ph.D. in Engineering Science and Mechanic

Yiming Niu

Dr. Sotiria Koloutsou-Vakakis, University of Illinois at Urbana - Champaign

Dr. Sotiria Koloutsou-Vakakis holds a Diploma in Surveying Engineering (National Technical University of Athens, Greece), a M.A. in Geography (University of California, Los Angeles), and M.S. and Ph.D. degrees in Environmental Engineering (University of Illinois Urbana-Champaign). She is a Senior Lecturer and Research Scientist in the Department of Civil and Environmental Engineering, at the University of Illinois Urbana-Champaign. Her main interests are in air quality, environmental policy, and supporting student learning and professional preparation.

Karthik Pattaje

Prof. Jacob Henschen, University of Illinois Urbana-Champaign

Professor Henschen completed his B.S., M.S., and PhD. at the University of Illinois Urbana-Champaign in 2007, 2009, and 2018 respectively. He was an Assistant Professor at Valparaiso University until he moved to the University of Illinois Urbana-Champaign as a Teaching Assistant Professor in June 2020. He serves as the co-chair for the Teaching Methods and Education Materials Committee at ACI and the co-chair of the Committee on Faculty Development at ASCE.

CASE STUDY: Integration of Python programming in a Civil Engineering laboratory course

John S. Popovics, Yiming Niu, Sotiria Koloutsou-Vakakis, Karthik Pattaje, Jacob Henschen

Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign, 205 N. Mathews, Urbana, IL 61801

1. Introduction

In this case study, we present an example of integrating Python programming assignments in a laboratory-based Civil Engineering (CE) course. As well established in the literature, the integration of programming and coding into discipline-specific engineering education is essential to address the growing demand for computational proficiency in engineering disciplines. The benefits to students of coding integration include enhanced problem-solving skills; a deeper understanding of engineering concepts through visualization, simulation, and modeling; and increased employability [1]. Implementation of such efforts also help satisfy ABET student outcomes such as (i) identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics; and (ii) acquire and apply new knowledge as needed, using appropriate learning strategies. Computing/coding platforms like Python [2] and R [3] can simplify the complexity of mathematical modeling and algorithmic problem-solving. Coding allows students to engage with complex systems in ways that were previously inaccessible, fostering active learning and engagement [4]. Despite these benefits, implementation challenges remain in non-computer science disciplines, including student fear of coding that are accentuated by disparities in prior programming knowledge among students, and sometimes faculty reluctance to take on such tasks that may be out of their comfort zone.

The redesign of laboratory assignments to include coding for data analysis and visualization is part of an effort within our department to thread coding, computing and computational thinking throughout our CE curriculum. The departmental initiative and the scholarly framework for this case study are presented in [5]. The goal is to integrate high-level interpreted programming languages into problem solving in all courses, thus boosting student computational and computational thinking skills, all within the CE discipline. Our students are first introduced to Python and R in required 1st and 2nd year courses. The next important step is to continue practicing and building upon these skills in upper-level courses so that students become confident users of these tools in problem solving, data analysis and visualization.

The case study course presented in this paper is a required 3rd year CE laboratory course on the behavior of materials. The course is also designated as an advanced composition course, aiming to sharpen student technical communication skills. Here, we use one laboratory as an example to present our approach for integrating Python for the compilation, presentation, and analysis of collected laboratory data presented in weekly laboratory reports.

2. Course learning objectives and objectives of computational tools

The learning objectives of the materials behavior laboratory course are to:

1. Understand physical and chemical material properties,
2. Conduct laboratory experiments,

3. Analyze experimental data,
4. Evaluate material performance, and
5. Develop technical written communication skills.

The laboratory testing generates large data files that require students to manually enter and process the data. These operations are repetitive from one laboratory experiment to the next which makes them ideal to automate with some computational process. Through the updated assignments that require Python coding, students see how programming can be used as a practical tool in their coursework. The updated assignments simultaneously support course objectives 3, 4, and 5.

For computation integration, we opted to employ Python [2] in the Google Colaboratory (Colab) [6] environment. In addition to CE students, the course serves Mechanical Engineering students. Students from both disciplines are introduced to Python in CS 101, which is a required prerequisite course, offered by the Computer Science (CS) department for non-CS students. Students subsequently take required 2nd year courses where Python is used. Thus, Python was well suited for the whole class population. Google Colab was chosen because it offers a collaborative, cloud-based environment supporting code and text. Students can write and execute Python code without the need for complex local installations. Real-time collaboration and feedback are possible. Because the course is also a required course aimed at the development of technical communication skills for CE students, students can submit their analysis, interpretation, and discussion of their results all in the same environment.

3. Approach

The integration of Python into the course begins with Laboratory Assignment 0 (Lab 0). Lab 0 serves as a “pre-lab” that introduces the Google Colab environment and provides a review of basic Python syntax, data structures, and functions (Figure 1). The course includes a total of eleven weekly laboratory assignments that follow. To address the labor-intensive nature of compiling laboratory reports - especially ones with an emphasis on writing - this initial lab assignment guides students in developing their own functions, which can be reused in later reports for example Lab 3, which is described in the next section. Students are guided to use Python to create visually clear, informative, and properly formatted graphs and images, and to analyze the data appropriately. Then in subsequent assignments they are encouraged to reuse and build upon their graph codes and templates. This approach allows students to work more efficiently and independently as the semester progresses, reducing the risk of task overload from repetitive tasks. For this case study, we illustrate application in a more advanced example laboratory assignment in the following section.

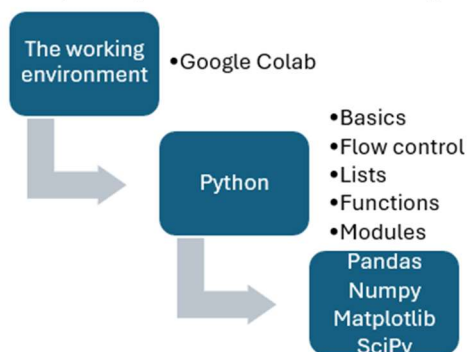


Figure 1. Summary of contents of review Lab 0.

3.1 Description of the example ‘Bending and Torsion Tests’ laboratory (Lab 3)

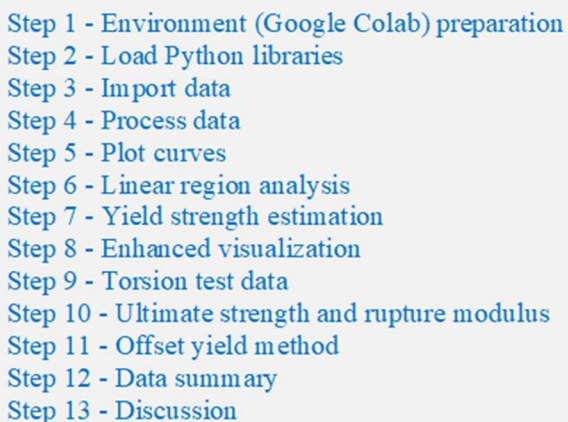
This Lab 3 assignment aims to investigate the mechanical behavior of structural materials, including steel, aluminum, and PMMA (polymethylmethacrylate), under bending and torsion loading. Students are guided to analyze and compare the elastic and plastic responses of materials and determine key properties such as the modulus of rupture and yield strength. The lab aims to provide insights into the similarities and differences between bending, torsion, compression, and tension behaviors, especially at failure. The laboratory activity comprises the following tasks:

- Conduct **four-point bending tests** on rectangular beam specimens using an Instron load frame to measure load and deflection,
- Perform **torsion tests** on circular cross-section specimens using a torsion machine to record torque and angular displacement,
- Measure material dimensions (e.g., width, depth, diameter) and collect experimental data using LabVIEW® software, and
- Analyze failure modes, create plots (e.g., load vs. deflection, torque vs. twist), and calculate mechanical properties using provided theoretical equations.

The Python assignments focus on and are required for the last task. Based on the collected laboratory data, Python is used to plot graphs showing load vs. deflection (bending) and torque vs. twist (torsion) responses and to analyze the material behavior including ductility, failure modes, and elastic-plastic transitions. The submitted assignments must include created plots, associated text with analyses, and the Python code (in the form of a Google Colab notebook) used to generate the plots and analysis all within a laboratory report setting. The appearance and correctness of plots and data count for 25% of the report grade and the Python code correctness counts for 5% of the grade.

3.2 Summary of guided Lab 3 steps

In Figure 2, we summarize the steps students are guided through using Python in the Google Colab environment, for Lab 3. Due to space limitations, it is not possible to include the full Lab 3 notebook here. Instead, we provide the link to the teaching assistant (TA) version of the Lab 3 demo workspace: <https://tinyurl.com/32awvmht>.



Step 1 - Environment (Google Colab) preparation
Step 2 - Load Python libraries
Step 3 - Import data
Step 4 - Process data
Step 5 - Plot curves
Step 6 - Linear region analysis
Step 7 - Yield strength estimation
Step 8 - Enhanced visualization
Step 9 - Torsion test data
Step 10 - Ultimate strength and rupture modulus
Step 11 - Offset yield method
Step 12 - Data summary
Step 13 - Discussion

Figure 2. Lab 3 - Bending and Torsion Tests - flowchart of steps on Google Colab.

In Figure 3, we present an example of grading rubric for a randomly chosen question from Lab 3 with a respective snapshot of a random student answer and grading comments. The question-specific rubrics help improve grading consistency among TAs.

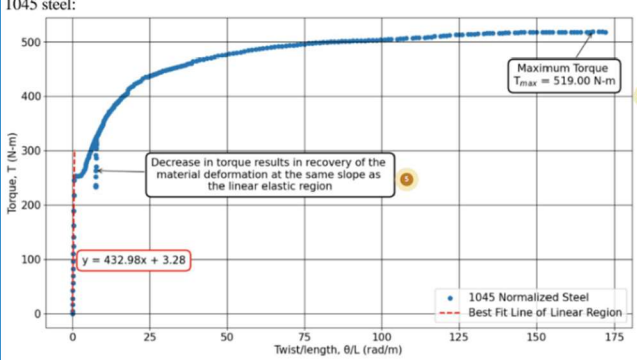
Question specific rubric	Example of student submission
<p>Q5 (8 pts)</p> <ul style="list-style-type: none"> – 0 pts Correct – 0.5 pts Few Graphing Issues – 1 pt Some Graphing Issues – 1.5 pts Several graphing issues – 5 pts missing graphs – 0.5 pts Incorrect calculation of G – 1 pt Incorrect calculation of E – 0.75 pts Missing G – 0.75 pts Missing E – 0.25 pts Minor mistake – 1.5 pts Missing Calculations – 8 pts Missing – 0.25 pts PMMA is strain-rate dependent – 0.5 pts error – 0.5 pts missing discussion 	<p>5. (8 pts) Using the data collected from the torsion tests, draw a diagram of torque T versus twist θ_L for each material.</p> <p>Estimate G, knowing J (Eqn. (15)) and the measured initial slope of the curve.</p> <p>Then, using a Poisson's ratio value of $\nu = 0.30$ for steel, 0.33 for aluminum and 0.40 for PMMA, back-calculate a value of Young's modulus E using the isotropic elastic relation G. Put the result for E in Table 2 and Table 3.</p>  <p>Figure 6: Torque vs Twist Curve for 1045 Hot-rolled Steel Under Torsion</p> <p>– 0.25 pts Few Graphing Issues</p>

Figure 3. Left panel: rubric for example question 5 of Lab 3. Right panel: example student submission for question 5. The numbered dots represent encoded comments. For the encoded comments shown in this example student submission, 5: avoid description inside graphs; 6: use sherif type font.

4. Discussion

Over the past few years, Python coding has been integrated into the course in stages, which gave us the opportunity to be deliberate and cautious with implementation and to obtain student feedback to make informed corrections. In the initial semester, only the first three labs (including Lab 0) were assigned to a subset of students who volunteered to use Python. The Lab 0-3 data was also similar between the various tests, involving stress and strain or load and deflection. This allowed students to develop a single algorithm that only needed slight modifications to accommodate the data going from Lab 0 to Lab 1 etc.

In the subsequent two semesters, all laboratory assignments were re-designed to integrate Python use and students were given the choice between completing their assignments with Python or with Excel. Starting this current semester (results not reported here) Python is required for all students and for most laboratory assignments. Throughout, we have obtained student feedback through surveys at the beginning and end of each semester.

In Fall 2024, 89 students filled the beginning of semester survey, and 81 students filled the end of the semester survey. When asked which tool they feel more confident using, Excel was the

tool most students feel more confident using, with Python being second. Focusing on Python, the percentages of students expressing confidence for Python were 66.3% and 81.5% in the early and end of semester surveys, respectively, with the difference being statistically significant at 5% confidence level ($p\text{-value} = 0.025$).

At the end of the semester (Figure 4), 52% of responding students answered ‘Yes’, 39% responded ‘No’, 9% did not respond to the question if they used Python for their assignments. Among the students who used Python, 93.5% expressed confidence in using Python compared to 57.1% who expressed confidence among those who did not use Python, which is a statistically significant difference ($p\text{-value} = 0.16 \times 10^{-3}$).

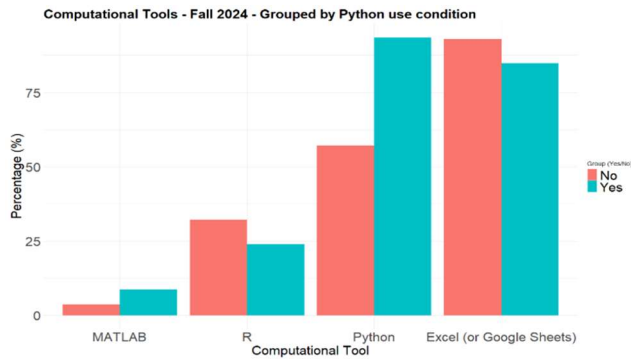


Figure 4. Self-reported student confidence in using different computational tools. Responses to the question “Which computational tools do you feel confident using? (check all that apply)” ($n=81$). Responses conditioned on using (Yes) or not (No) Python for the assignments.

The graph in Figure 5 displays responses to the question ‘How well integrated was the coding component with the other topics in this course?’. Of the 28 students who answered this question and who had not chosen Python (“No”) 54% responded “Poorly Integrated” and 32% “Neutral”. Of the 46 students who had chosen Python (“Yes”) 35% responded “Neutral”, 50% “Well Integrated” and 6.5% “Very well integrated”.

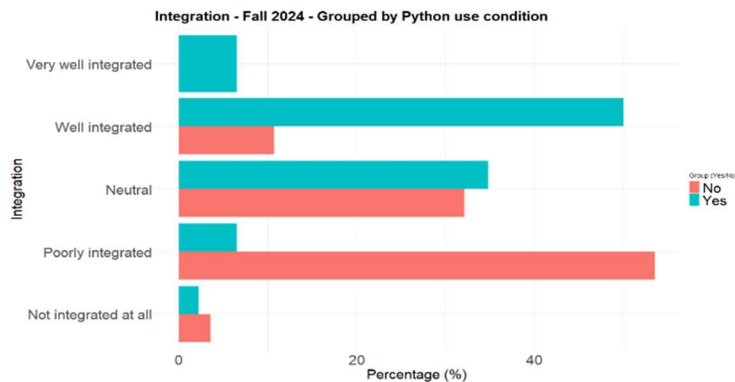


Figure 5. Student end of semester feedback regarding integration of coding in the course, conditioned on if students chose Python (Yes) for their assignments. Question: “How well integrated was the coding component with the other topics in this course?” ($n=74$).

5. Conclusion

The integration of coding and computational tools in engineering education is transforming how students engage with the discipline. Despite challenges such as knowledge disparities, non-negligible learning curves, and initial resistance to change, the benefits - ranging from enhanced understanding of concepts to industry readiness - the importance of adopting these tools and technologies is underscored. Tools like Python and Google Colab combine accessibility and collaboration that can further enrich the educational experience, preparing students for the

digitally driven engineering landscape. Despite initial hesitation, student feedback indicates that most students respond positively to the new requirements when they engage with the platform.

Acknowledgments

This work is supported by a 3-year (2022-2025) grant from the Strategic Instructional Initiatives Program (SIIP) of The Grainger College of Engineering and matching funds by the Department of Civil and Environmental Engineering, at the University of Illinois Urbana-Champaign. We thank the Grainger College of Engineering Academy for Excellence in Engineering Education (AE3) Education Innovation Fellows who mentored us in different phases of this project: Professors A. Schleife, A.A.M. Alawini, and C. Radhakrishnan.

References

- [1] A. Richert, M. Shehadeh, L. Plumanns, K. Groß, K. Schuster and S. Jeschke, "Educating engineers for industry 4.0: Virtual worlds and human-robot-teams: Empirical studies towards a new educational age," *2016 IEEE Global Engineering Education Conference (EDUCON)*, Abu Dhabi, United Arab Emirates, 2016, pp. 142-149, doi: 10.1109/EDUCON.2016.7474545.
- [2] G. Van Rossum and F. L. Drake, "Python 3 Reference Manual". Scotts Valley, CA, USA: CreateSpace, 2009.
- [3] R Core Team, "R: A Language and Environment for Statistical Computing". Vienna, Austria: R Foundation for Statistical Computing, 2021. [Online]. Available: <https://www.R-project.org/>.
- [4] J. C. Perrenet, P. A. J. Bouhuijs, and J. G. M. M. Smits, "The suitability of problem-based learning for engineering education: Theory and practice," *Teaching in Higher Education*, vol. 5, no. 3, pp. 345–358, 2000. [Online]. Available: <https://doi.org/10.1080/713699144>.
- [5] S. Koloutsou-Vakakis, M. L. Matthews, C. Cohen, J. Henschen, J. S. Popovics, Ashlynn S. Stillwell. "Surveying civil engineering student attitudes toward the use of computational tools". Civil Engineering Division., 2025 ASEE Annual Conference & Exposition, Montreal, Canada [Accepted].
- [6] Google, "Google Colaboratory," Dec. 20, 2024. [Online]. Available: <https://colab.research.google.com>.