9th International Research Symposium on Problem-based Learning (IRSPBL):

TRANSFORMING ENGINEERING EDUCATION 2023

Edited by: Aida Guerra, Juebei Chen, Rea Lavi, Lykke Bertel, and Euan Lindsay

Harvard John A. Paulson School of Engineering and Applied Sciences

United Nations Educational, Scientific and Cultural Organizatio

AALBORG UNIVERSITY Aalborg Centre for Problem Based Learning in Engineering Science and Sustainability under the auspices of UNESCO

MIT School of Engineering

NEET New Engineering Education Transformation

Transforming Engineering Education
Edited by Aida Guerra, Juebei Chen, Rea Lavi, Lykke Bertel, and Euan Lindsay

Series: International Research Symposium on PBL

© The authors, 2023

Graphic design by Hjortlund Medier

ISBN: 978-87-7573-023-0
ISSN: 2446-3833

Published by Aalborg University Press I forlag.aau.dk

9th International Research Symposium on PBL, June 21-23, 2023
Transforming Engineering Education

Hosted by MIT School of Engineering, Harvard John A. Paulson School of Engineering and Applied Science, and co-organised with Aalborg PBL Centre for PBL in Engineering Science and Sustainability under the Auspices of UNESCO, Aalborg University.

All the IRSPBL proceedings are available at https://aauforlag.dk/ and https://www.ucpbl.net/global-network/research-symposia

Harvard John A. Paulson School of Engineering and Applied Sciences

United Nations Educational, Scientific and Cultural Organization

AALBORG UNIVERSITY
Aalborg Centre for Problem Based Learning in Engineering Science and Sustainability under the auspices of UNESCO

MIT School of Engineering
NEET New Engineering Education Transformation

Work in Progress: Integrating Python into Mechanical Engineering undergraduate curriculum

Lama Hamadeh
University College London, United Kingdom, l.hamadeh@ucl.ac.uk

Summary

Integrating the fundamentals of computer science and programming skills into the undergraduate engineering curriculum has been a primary focus for many educational institutions around the world. Learning the basics of programming from the beginning of undergraduate engineering education allows students to incorporate such skills into their work in the future with ease. The department of mechanical engineering at University College London has acknowledged this value and decided to implement a programming element into the first-year mechanical engineering curriculum to teach the basics of Python language and assess it using a real-life engineering problem. Python is general-purpose, concise, easy-to-read and -learn programming language that has become one of the most popular and in-demand languages in the world. Python has a vast ecosystem of tools, packages, and libraries that address a wide-ranging number of programming scenarios and provide mechanical engineers with a large array of general-purpose functionality. The addition of this element to the first-year curriculum during the last academic year 2021-2022 has shown a high assessment passing rate and notable student engagement. In this extended abstract, an overview of planning, implementing and the results obtained from this process will be illustrated, and future work plans will be outlined.

Keywords: Python, mechanical engineering, computer science, undergraduate curriculum, programming skills.

Type of contribution: Best practice extended abstracts

1 Introduction: Python for Mechanical Engineers.

Programming skills provide engineers with an opportunity to integrate innovative technologies into their everyday work and make it more efficient. Even the knowledge of one programming language or understanding of how to work with data gives engineers a substantial advantage in their work. For the aim of giving rise to highly equipped engineering graduates, teaching these skills needs to be considered and implemented from the very beginning of their undergraduate university journey. Learning how to code allows students to start thinking like a programmer and improve their problem-solving abilities (Oliphant, 2007). After all, both engineering and programming include multiple tries and attempts to develop high-quality results, so with experience, they will start finding solutions for these problems much easier.

Choosing the right introductory programming language to teach undergraduate mechanical engineering students should be based on several factors, e.g., its simple syntax that makes it easy to learn for all beginners, its adaptability to analyze and explain most of the mechanical engineering problems, and its popularity within the computational communities, industry, and academia, which greatly helps with students' employability. For example, MATLAB is a widely taught software in most undergraduate STEM subjects that are used to write code for solving assignments, plotting graphs, and data analysis (Liu, 2020; Mueller, 2003). In addition, C language is especially useful for mechanical engineers because it is the language of choice for hardware interfaces, and is commonly used for microcontroller data acquisition and real-time robotic control (Furman et al, 2010; Salzman et al, 2013; Rehberger et al, 2013). Python is another high-level language and at first sight very similar to MATLAB: it is interpreted, has an interactive prompt, allows dynamics typing, and provides automatic memory management (Raymond, 2008; Nanz & Furia, 2015; Fangohr, 2004; Manish, 2021; Kumar et al, 2020). A comparison between these three programming languages in the context of teaching in engineering has been studied in (Fangohr, 2004). It has been found that Python is selected to be the best choice in terms of clarity and functionality of a programming language that provides engineering students with clear, unambiguous, and intuitive syntax that allows them to express their algorithms quickly. Python has a small core of commands, which provides nearly all the functionality beginners will require. Additionally, its vast ecosystem of tools, packages, and libraries addresses many programming scenarios and provides engineers with a large array of general- and special-purpose functionality. Most interestingly, Python has seen rapid growth as an introductory language in computer science courses and is becoming one of the most popular programming languages in industry and academia (Wende et al, 2020; Davim et al, 2019). When Python was released in 1991, the premier languages at the time were FORTRAN, COBOL, C, and C++. Since the mid-90s, it has steadily been increasing in popularity and overtaking its old competitors as its programs tend to be much shorter than equivalent programs in other languages. Learners' rates have also skyrocketed due to the fact that Python is an open-source language, meaning that anyone can contribute to the code. This surely makes Python stands out from all the other introductory programming languages. Fig 1 shows the percentage of questions received by Stack Overflow, a public question-answer platform for developers, from Python users over roughly 10 years. It can be clearly seen how the interest in Python has noticeably increased compared to other programming languages (Stack Overflow, 2023).

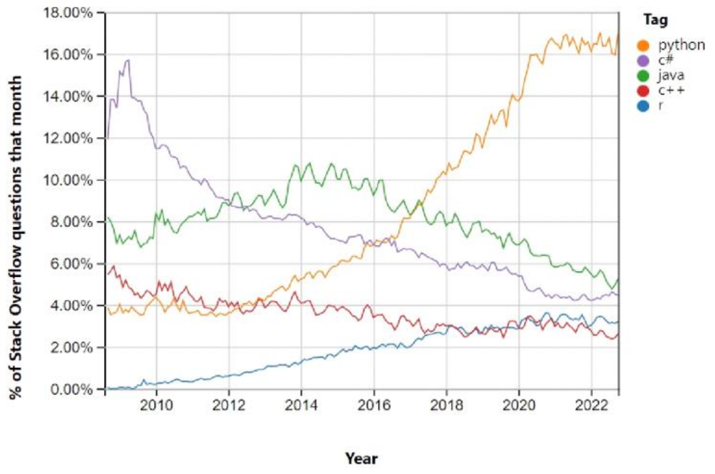

Figure 1: A snapshot of how programming languages have trended over time based on their tags on Stack Overflow since 2008 (Stack Overflow, 2023).

This has given motivation to some academics to transition to Python in their mechanical engineering teaching courses (Furman et al, 2020). As a result, a decision has been consensually made to implement Python in this element since it encapsulates largely all the pre-mentioned factors an introductory programming language must meet for teaching first-year mechanical engineering undergraduates.

2 Framework for a Best Practice

The aim of introductory courses in the first year of mechanical engineering is to provide students with the engineering fundamentals and show how they are applied to basic real-life problems. When it comes to designing an introductory programming element and integrating it into the first-year curriculum, it must go in line with this aim (Sheth et al, 2016). The content, objectives, and the way everything is learned, taught, and assessed need a lot of thought to be successful (Sobral, 2021). For this introductory programming element, and since no prior programming experience is required, the content and its associated activities should be progressive, i.e., starting from the most fundamental with simpler questions and advancing gradually toward more complex concepts (Gomes & Mendes, 2009). This hierarchical structure means that in order for the students to be able to explain and analyze a given engineering problem using Python, they must attain prerequisite **knowledge** (understanding the principles of algorithmic thinking and the language syntax), and **skills** (implementing Python commands *correctly* while making use of its available libraries), in the course of a certain teaching and learning approach. Furthermore, it is easier to achieve goals when they are well-defined. The clear and structured definition of instructional objectives, considering the acquisition of knowledge and skills, will direct the teaching process towards the appropriate choice of strategies, methods, delimitation of specific content, and assessment instruments, and, consequently, effective and lasting learning (Sobral, 2021). For the aim of designing an introductory programming element and creating clear learning objectives, the levels of the revised Bloom's taxonomy (Krathwohl, 2002) are chosen to be the drive for creating a holistic teaching and learning plan. This taxonomy has been proven to be effective in formulating undergraduate introductory programming content (Sobral, 2021; Gomes & Mendes, 2009; Britto & Usman, 2015) and assessments (Chatzopoulou & Economides, 2010). The revised Bloom's taxonomy is a hierarchical two-dimensional framework: Knowledge and Cognitive Processes with six levels: remember, understand, apply, analyze, evaluate, and create, as shown in Fig 2. In this element, and as shown in Fig 2, a mapping to each level of the revised Bloom's taxonomy has been drawn to its design and its underlying objectives. The first four levels correspond to the teaching and learning classes that deliver the main programming materials and ultimately build foundational knowledge and skills. A blend of both the fourth and fifth levels is directed toward assessing students' comprehension of the programming material. The final level, i.e., creating, is targeted at higher undergraduate years where students can implement and advance their Python knowledge onto new engineering projects.

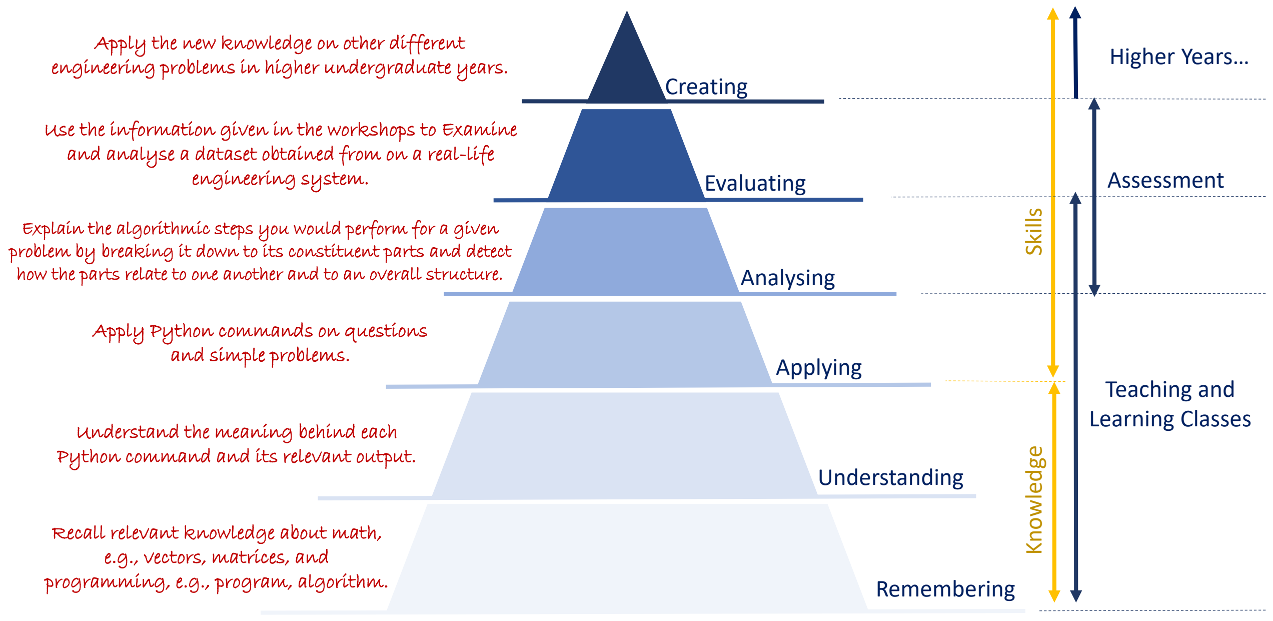
Apply the new knowledge on other different engineering problems in higher undergraduate years.
Creating
Use the information given in the workshops to Examine and analyse a dataset obtained from on a real-life engineering system.
Evaluating
Explain the algorithmic steps you would perform for a given problem by breaking it down to its constituent parts and detect how the parts relate to one another and to an overall structure.
Analysing
Apply Python commands on questions and simple problems.
Applying
Understand the meaning behind each Python command and its relevant output.
Understanding
Recall relevant knowledge about math, e.g., vectors, matrices, and programming, e.g., program, algorithm.
Remembering
Skills
Knowledge
Higher Years…
Assessment
Teaching and Learning Classes

Figure 2: Revised Bloom's Taxonomy projected onto the programming element introduced for the first-year mechanical engineering undergraduate course.

To systemize the best practice, the plan, as shown in Fig 3, starts with four teaching and learning classes where all the material is delivered in live-coding sessions depicting the first four levels of the taxonomy, followed by an assessment that tests students' apprehension and their ability to analyze and evaluate a given mechanical engineering system.

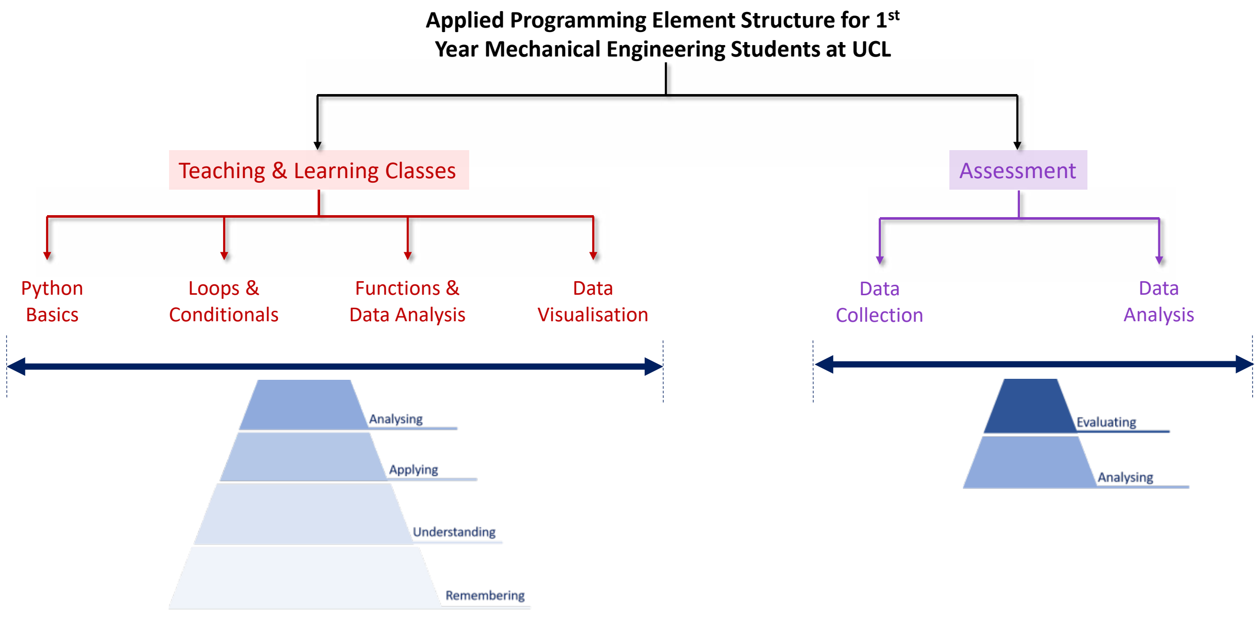

Figure 3: Teaching and Learning plan for the Introductory Python Programming Element.

2.1 Teaching and Learning Classes

In order to incorporate the first four levels of the pyramid into the students' programming learning journey, i.e., remembering, understanding, applying, and analyzing, it is essential to structure the programming classes so that they enhance students' engagement with the material, enable students to receive instant feedback and stimulate peer-to-peer discussion. Another aspect that must be considered is that a variation in programming abilities and skills between students should be expected. Some students will enter with a certain programming background, whilst the majority are novices in this field (Cawthorne, 2021). Assuming that all students have no prior programming experience, and for the aim of creating inclusive and active coding classes, four, 2-hours, in-person, live coding workshops were scheduled for the entire cohort that has roughly 200 students to be delivered throughout four weeks, i.e., one workshop per week. During these workshops, students are strongly advised to attend with their laptops so they can practice and develop their coding-writing skills and master the syntax of the language. Moreover, to provide programming material that is suitable for all students, the fundamentals of Python were chosen to be delivered; Python basics, iterative loops, conditionals, functions, working with data, and visualization. After explaining each topic and its required syntax, several problems are followed so students can solve using the correct concept. This is essential as it builds the required skillset that allows students to break down a larger task into smaller blocks that can be performed by code (Cawthorne, 2021). To manage the large number of students and to make certain that all of them receive the necessary feedback while they solve and code the questions, four postgraduate teaching assistants and I are constantly roaming around the lecture theatre to make sure students' questions are answered and their learning journey is on track.

2.2 Assessment

A typical introductory programming course will assess students by having them complete coursework so students are assessed on their ability to apply programming to solve problems (Cawthorne, 2021). For this element's assessment, and since it is directed toward first-year mechanical engineering students, it is designed so that it examines and measures the students' practical and programming skill sets. In this assessment, *Students are required to collect a dataset from a 3D-printed Stirling Engine and use it to answer coursework questions that analyze its physics, dynamics, and kinematics in a single Jupyter Notebook using*

Python language and submit it into the online associated submission portal. Stirling Engine is a mechanical engineering system that uses cyclic compression and expansion of a gas at different temperatures to convert heat energy into mechanical work at a certain frequency. The collected dataset, which has the format of a .csv file, includes parameters such as the engine's running time, the upper and lower temperatures, the number of revolutions, etc. Students are required to use this dataset to answer questions presented to them as coursework with the aim of evaluating and quantifying the physics, dynamics, kinematics, and efficiency of the engine. Their coursework answers must be written and submitted as a single Jupyter Notebook file. This would surely allow evaluating objectively students' technical skills and scientific knowledge.

2.3 Next Steps.

The top of the revised Bloom's pyramid is reserved for "creating". For this element, the content of Python programming provided in Year 1 is linked to skills in data analysis and the visual presentation of graphics. This builds the foundational knowledge and essential skills that are helpful in subsequent years when higher-level programming is necessary.

- In Year 2, the students primarily use data analysis and visual presentation skills in laboratories, such as Aerofoil testing (in Intermediate Fluid Mechanics), beam buckling (Solid Mechanics), and strain gauge measurements (Instrumentation).
- In Year 2, the above skills are used in laboratories related to Advanced Dynamics and Control. Importantly, the students undertake a significant year-long individual research project, where they apply advanced data analysis. Several projects require advanced programming skills, e.g. in machine learning (ML), parametric model setup, and artificial intelligence. The success of these projects was limited in the past because the students lacked knowledge of basic programming, and this was fed back to the curriculum development team on multiple occasions.
- In Year 4, the students undertake a year-long group design project that is worth 50% of the year's credits. There is always a requirement for advanced data analysis in these projects. the students also undertake complex projects that involve building and testing ML platforms, creating advanced codes for image recognition, etc., where advanced knowledge of programming languages is required. These students are much better equipped now that the fundamentals of Python are acquired in Year 1.

3 Outcomes, Conclusion, and Future Plans

Introducing the Python programming element to the first-year students of a cohort of roughly 200 students last academic year, 2021-2022, has shown significant success. This can be evaluated based on two factors:

- **Coursework results**. As mentioned in 2.2, students had to submit a single Jupyter Notebook file that contains the answers to coursework questions that analyze the dynamics of a Stirling Engine. The marking criteria have been set out in a way that they depend on three main aspects: quality of code, the use of markdown cells, and the accuracy of scientific content. Grades are split into four bands: A (excellent), B (well executed), C (competent), D (marginally accepted), and F (fail). Fig 4 shows the coursework grades distribution where it can be seen that 94% of students passed the coursework with the 44% majority falling into the B (well executed) grade. It is worth noting here that the dominant cause behind the 6% fail rate was mostly technical as most of these students were unable to save their Jupyter Notebooks properly (they used the "save as" option from the browser rather than Jupyter software), and as a result, markers were unable to open and see their submitted files. This issue has been stressed on it this year to prevent it from re-occurring.

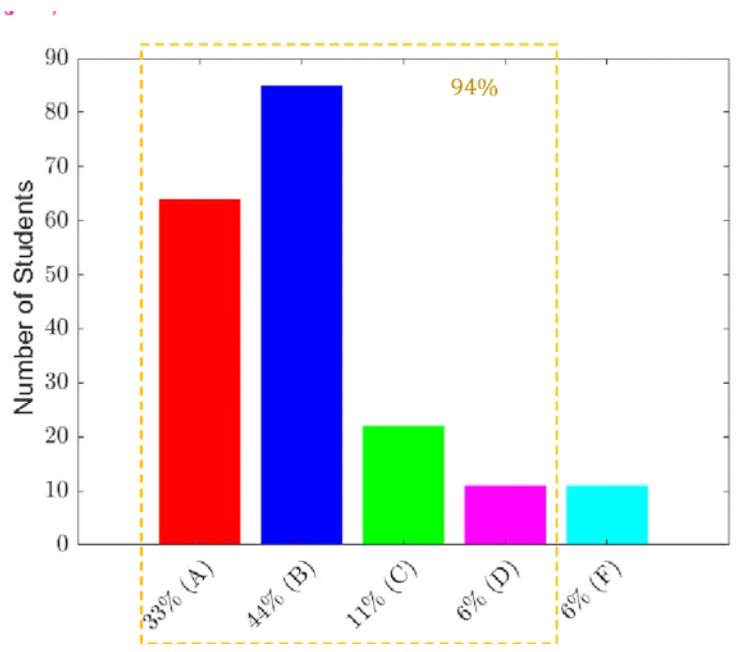

Figure 4: Grades percentages distribution of the programming assessment.

- **Students' motivation**. Throughout the entire teaching and learning journey of this element, students showed several behavioral signs that proved their motivation and interest in learning Python. The average workshop attendance was roughly 70%. This created vibrant and interactive learning classes where students tended to sit in groups that positively enhanced their collaborative discussion. Moreover, many students did not apply the material blindly during workshops; they engaged with the thought process and often proposed alternative algorithmic ways to solve questions. These are certainly good indications that students had an enjoyable introductory programming experience. Worth mentioning that for the existing later-year students who have not had the basic Python experience and showed interest in this introductory course as many emails were received asking for more details, all the material was shared with them so they can go through the content at their own pace.

In conclusion, integrating an introductory Python programming element into the first-year mechanical engineering curriculum has shown to be a necessary course of action. Putting together an effective instructional design for this element would certainly lay out the required foundational programming knowledge and skills that our students need for their academic progression in higher years and their future careers. Surely, and since this element is new and still in progress, several future ideas are needed to be thoughtfully considered so this learning activity is taught as smoothly and holistically as possible for the next years. For instance, the problems introduced during the active learning workshops that students use to solve computationally must be related to other modules taught in the course, e.g., thermodynamics, elasticity, solid body kinematics, fluid dynamics, etc. This will build a holistic element that not only equips students with the necessary programming skills but also allow them to project these skills on the topics taught in their courses. Furthermore, and as mentioned before, since higher-year students deal with engineering projects that require advanced programming and data analysis knowledge and skills, new elements that focus on algorithms and methodologies that address key tasks in data-driven engineering can be introduced and integrated suitably into relevant modules. This would provide students with additional expertise to not only analyze their projects effectively and extract useful insights from them but also add an auxiliary hard employability skill into their academic package.

4 References

Britto, R. & Usman, M., (2015). Bloom's taxonomy in software engineering education: A systematic mapping study. *IEEE Frontiers in Education Conference (FIE)*, El Paso, TX, USA, pp. 1-8.

Cawthorne, L. (2021). Invited Viewpoint: Teaching Programming to Students in Physical Sciences and Engineering. *Journal of Materials Science*, vol. 56, pp. 16183-16194.

Chatzopoulou D. I., & Economides A. A. (2010). Adaptive Assessment of Student's Knowledge in Programming Courses, *Journal of Computer Assisted Learning*, 26(4), pp 258-269.

Davim, J. P., Díaz Vicente García, & Solanki, V. K. (2019). Handbook of IoT and Big Data. *CRC Press*.

Fangohr, H. (2004). A comparison of C, MATLAB, and python as teaching languages in engineering. *Computational Science* - ICCS 2004, 1210–1217. https://doi.org/10.1007/978-3-540-25944-2_157

Furman, B. J., Ahsan, S., & Wertz, E. (2020). Making the move from C to Python with mechanical engineering students. *Paper presented at 2020 ASEE Annual Conference & Exposition Virtual Conference.*

Furman, B., & Wertz, E. (2010). A first course in computer programming for mechanical engineers. *In Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications* (pp. 70–75).

Gomes, A. & Mendes, A. (2009). Bloom's taxonomy based approach to learn basic programming. In G. Siemens & C. Fulford (Eds.), *Proceedings of ED-MEDIA 2009--World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 2547-2554).

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, vol. 41, no. 4, pp. 212-218.

Kumar, G., Singh, V., & Thombre, M. (2020). Importance of Learning Python Programming in the Field of Mechanical Engineering. *United International Journal for Research & Technology*, *1*(12), 16-18.

Liu, Y. C. (2020). Implementation of MATLAB/Simulink into a vibration and control course for mechanical engineering students. In *Proceedings of the ASEE SE Section Annual Conference* (pp. 8-10).

Manish, P. (2021). Exploring Integration of Python Libraries in Computation Intensive Mechanical Engineering Courses, *International Journal of Engineering Research & Technology (IJERT) ICDML* – 2020 (Volume 09 – Issue 02).

Mueller, D. (2003). Introducing the finite element method to mechanical engineering students using MATLAB. In *2003 Annual Conference* (pp. 8-781).

Nanz, S., & Furia, C. A. (2015). A comparative study of programming languages in Rosetta Code. 2015 *IEEE/ACM 37th IEEE International Conference on Software Engineering*. https://doi.org/10.1109/icse.2015.90

Oliphant, T. E. (2007). Python for Scientific Computing. *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10-20.

Raymond, E. S. (2008). The art of Unix programming: With contributions from thirteen Unix pioneers, including its inventor*, Ken Thompson. Addison-Wesley*.

Rehberger, S., Frank, T. & Vogel-Heuser, B. (2013). Benefit of e-learning teaching C-programming and software engineering in a very large mechanical engineering beginners class. *IEEE Global Engineering Education Conference (EDUCON)*, Berlin, Germany, pp. 1055-1061.

Salzman, N., & Meckl, P. H. (2013). Microcontrollers for Mechanical Engineers: From Assembly Language to Controller Implementation. *Paper presented at 2013 ASEE Annual Conference & Exposition*, Atlanta, Georgia. 10.18260/1-2—22290

Sheth, S., Murphy, C., Ross, K. A., & Shasha, D. (2016). A course on programming and problem solving. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*. https://doi.org/10.1145/2839509.2844594

Sobral, S. R. (2021). Bloom's Taxonomy to Improve Teaching-Learning in Introduction to Programming. *International Journal of Information and Education Technology*, 11(3), 148-153. DOI: 10.18178/ijiet.2021.11.3.1504. ISSN: 2010-3689. Disponível no Repositório UPT, http://hdl.handle.net/11328/3368

Stack Overflow is a question and answer website for professional and enthusiast programmers. Stack Overflow Trends show how technologies have trended over time based on use of their tags since 2008, when Stack Overflow was founded. https://insights.stackoverflow.com/trends

Wende, M., Giese, T., Bulut, S., & Anderl, R. (2020). Framework of an active learning python curriculum for First Year mechanical engineering students. *2020 IEEE Global Engineering Education Conference (EDUCON)*. https://doi.org/10.1109/educon45650.2020.9125259