A Python-based lab module to conduct thermodynamic cycle analysis

Alexa Aulicino
Department of Mechanical Engineering
Rowan University
Glassboro, NJ, USA

Smitesh Bakrania, Ph.D.
Department of Mechanical Engineering
Rowan University
Glassboro, NJ, USA
bakrania@rowan.edu

Abstract*—A Python-based lab activity was developed for a remote Thermodynamics course to add computational thinking to traditional analytical problem solving. Python was selected to conduct the analysis because of its popularity and utility in the broader engineering field. Early exposure to this highly desired engineering skill can provide added benefits to students. Combining Python with an engaging lab experience can have a compounding effect on student learning outcomes. A five-week Python-based lab module was developed for an introductory thermal-fluid science class. The module reinforced fundamental concepts learned in lecture, while expanding on design-related analysis which is often left for advanced courses. The lab module began with an introduction to Python programming and quickly transitioned to the parametric analysis of standard Rankine, Gas Turbine, and Vapor Compression cycles. The lab module was designed to be self-guided with step-by-step instructions presented using Google Colab. This paper details the implementation and the student outcomes. Both direct and indirect assessments were conducted over two semesters of the course. Results indicate a strong positive impact on Python programming learning outcomes. Students acquired a working knowledge of Python programming and experienced how computational tools can be used to solve advanced engineering problems. At the same time, the student feedback indicated students' resistance to open-ended projects and independent learning; even if they are aware of their relevance and benefits to their future careers. Nevertheless, the positive learning outcomes were encouraging. Whether students pursue a career in thermodynamics or in a broader engineering field, this lab experience equipped them with tools that can augment their engineering skills.

Keywords*—Thermodynamics, Python, Computational Thinking, Open-ended, Online Learning

I. INTRODUCTION

Thermodynamics serves as a fundamental engineering course for thermal-fluid sciences before diving into fluid mechanics and heat transfer. Often the homework assessments involve close-ended problems. In this capacity, students are able to self-assess and practice but are not challenged with complexity that they may experience professionally. Professional engineering problems are often more variable and open-ended [1][3]. Engineers must consider constraints, recognize unknown variables, weigh various options, and conduct appropriate analysis before making the best decision. Additionally, students show strong retention when performing design and analysis when compared to single-answer problems [4]. Since design is an iterative process, even a rudimentary open-ended component incorporated into a traditional problem begs a deeper grasp of course content [5]. By conducting parametric analysis of complex systems, students can practice their engineering decision making skills, thus advancing thermodynamic fundamental comprehension. In addition, computational thinking when applied to courses besides computer science courses is developmentally critical for engineering students [6]. While in-person laboratory exercises can be a powerful vehicle to deliver such an experience, very few projects exist to suit the transition to remote learning. The combination of open-ended and computational problems can greatly expand student learning outcomes.

Projects that use computational tools are particularly helpful, because they reinforce fundamentals through critical engineering tools. Besides, integrating computational thinking in engineering courses can serve to broaden the skills and capabilities of our engineers [6], [7]. Computational thinking expands on traditional analytical processes by using methods common with computer scientists to solve complex problems [8]. Often these data intensive methods can be applied to a variety of engineering problems. In the past, there have been several examples of thermodynamic projects that used computational tools like MATLAB [9], Microsoft Excel [10], and Engineering Equation Solver (EES) [4]. These tools are already common within the Mechanical Engineering discipline. However, Python serves as a potent alternative, growing in popularity for data analysis [11]. A number of attempts have been made to add Python as a platform to teach programming to engineering students [12][17]. However, these are confined to introductory courses rather than integrated into core engineering courses. Beyond teaching the fundamentals of programming, engineers must be able to translate an engineering problem into a computational problem. Core engineering courses are ideal for students to practice their computational thinking in conjunction with their engineering practice.

Considering the en masse transition to online instruction and the limited computational tools accessible by remote students [11], a Python-based lab module for a thermodynamic course was developed to explore thermodynamic cycles and engineering decision-making. The implementation afforded

incorporation of computational thinking into thermodynamics and, at the same time, enhanced the remote learning experience. The lab exercise required students to learn Python and then use it to conduct parametric analysis on advanced thermodynamic cycles. The exercise was designed under the assumption that the students did not have working knowledge of Python. The module was self-paced with the inclusion of explanatory pre-recorded videos. The lab modules can be integrated into an existing thermodynamics course over the last five weeks of the term. In this paper, the broad implementation and the project outcomes are presented. The implementation details can be obtained directly from the authors. Both direct and indirect assessments are documented to show the positive outcomes specifically related to computational thinking for the students. The design of the lab module is such that it can be easily adopted into any existing thermodynamics course, whether remote or in-person. More importantly, Python can be integrated into courses besides Thermodynamics to infuse computational thinking broadly. The project carries minimal increase in teaching load. The key benefit of this implementation is that the project not only allowed the thermodynamic fundamentals to be reinforced, the students also acquired a highly desired engineering skill.

II. Development and Implementation

The cloud-based Google's CoLaboratory (Google Colab) environment was selected for this exercise. Google Colab allows Python code execution within a browser and uses the notebook structure to provide textual context to the code. Python requires specific libraries to be used for plotting, mathematics, and, more pertinent, to access thermodynamic properties. The PYroMat thermodynamic properties library was selected for use with this lab module. PYroMat was developed by Martin, et al. at Pennsylvania State University [18], [19]. It allows users to retrieve thermodynamic properties from almost 1,000 substances. While there are other alternatives [20], PYroMat was a better fit for the purposes of this project.

The module was designed to be self-paced with step-by-step instructions provided for each exercise. It was assumed that this was the first-time students will be using Python. Video tutorials were produced for the introductory sections for students to become familiar with the programming environment. This asynchronous approach was used to match the instructional style of the accompanying remote-learning course, which was also taught asynchronously. The students had five weeks to complete the entire lab-module. The students had the choice to complete each exercise concurrent with the material being covered or wait until all class material was presented. The lab-module incorporated five different thermodynamic cycle analysis and were mapped to the course textbook by Dr. John Reisel [21]. The included open-ended exercises required students to iterate on the parameters and predict cycle performance. The entire lab-module was presented as a single Google Colab notebook housing each of the five exercises. Students worked through each sequentially before moving on to the next exercise. Table I presents the exercises in terms of deliverables and the associated thermodynamic learning outcomes. Each exercise provided engineering context, background, and the requisite parameters to begin the analysis. Within the Colab notebook, space was provided to include the program solution and the graphical output for each exercise. An attempt was made to present realistic cycle parameters of existing power plants in the region. Grading rubrics were supplied to the students along with clear instructions related to completing the lab-module. It was estimated that the project involved 20-30 hours of student effort.

TABLE I. The Table Presents the Five Exercises Included in the Lab Moudle. Each Exercise is Associated with a Numbered Deliverable and the Corresponding Learning Outcome (LO) Statement that Begins with, "Students will be able to…".

No.	Deliverables and Learning Outcomes
1	*1.1 Property Retrieval* LO: … use Python to retrieve thermodynamic properties
2	*2.1 Rankine: Efficiency vs. Peak Pressure* LO: … study the impact of peak pressure in a basic Rankine cycle
	2.2 Rankine: Peak Superheat Temperature LO: … study the peak temperatures and power output relationship
3	*3.1 Non-ideal turbines* LO: … determine the impact of non-ideal turbines on net power
	3.2 Reheat stages LO: … study the role of reheat stages on a non-ideal Rankine cycle
4	*4.1 Brayton Cycle Performance and the Peak Pressure* LO: … study how peak pressures impact Brayton Cycle performance
	4.2 Combined Cycle LO: … compare Rankine, Brayton, and a Combined Cycle outputs
5	*5.1 Refrigeration Cycle Design* LO: … compare cycle performance using three different refrigerants.

Exercise 1 asked students to retrieve the missing properties for the provided state for water and R134a. Exercise 2 focused on the Rankine cycle and required students to study how the peak cycle pressure and temperature impacted performance. Exercises 3 and 4, similarly asked students to study Rankine and Brayton cycle modifications, respectively. The final exercise was related to the vapor compression cycle and comparing various refrigerants for a specific application. All exercises forced students to develop trends and evaluate the performance with plots generated within Python.

III. Assessing Student Perceptions and Outcomes

In order to effectively assess the learning outcomes of this new lab module, both direct and indirect assessment methods were used. The learning outcomes were related to the computational aspects, as opposed to thermodynamics, considering the novelty of the Python integration. Therefore, the direct assessment used pre- and post- questions to test Python fundamentals. Direct assessment of the thermodynamic concepts was not explored to avoid the compounding effect of the lectures versus the lab exercise in a remote setting. The indirect assessments looked at student perceptions before and after completing the lab module. The assessments were conducted using online survey forms that were anonymous.

The module was implemented within an existing thermodynamics course taken by mechanical engineering undergraduate students in their sophomore year or prior to being juniors. To ensure participation in the survey, students completed the survey before accessing the project details and finally before submitted the project. The study was conducted over two semesters in 2021: Spring and Summer. Both courses were designed to be delivered asynchronously via Canvas LMS. The Spring 2021 course had 33 students enrolled and the Summer 2021 course had 18 students enrolled. The Spring 2021 course spanned 16 weeks, whereas the Summer 2021 course

spanned only 8 weeks. As stated earlier, both courses were delivered asynchronously to students who were not on campus due to the pandemic restrictions.

A. Direct: Python Learning Outcome Assessment

The Python Learning Outcome Assessment consisted of a fundamental Python knowledge test related to coding. There were 10 multiple choice questions with topics that included code output, arrays, list comprehension, library import and usage, and graphing. The questions were mapped to the project learning outcomes. The same questions were administered before and after the lab module was completed to assess prior knowledge and what the students gained from the project. The assessment was ungraded, and anonymous.

B. Indirect: Student Perceptions Questionnaire

The Student Perception Questionnaire consisted of two components: a set of Likert scale rating questions as well as short answer or open-ended questions.

1) Rating Questions

The nine Likert scale rating questions are listed in Table II. The questions can be divided into three parts: (a) student comfort with programming and Python, (b) their attitude towards open-ended and independent work projects, and (c) their preference for Python as an analytical tool. For instance, the students were asked about their likelihood of using Python for future engineering problems.

TABLE II. RATING QUESTIONS USED TO ASSESS STUDENT PERFECPTIONS OF PROGRAMMING WITH PYTHON. RATINGS SCALED FROM 1 FOR "LOW" TO 5 FOR "HIGH" RESPONSES.

No.	Question Statement
1	Your comfort-level with programming.
2	Your prior awareness of Python.
3	Your prior experience with Python.
4	Your comfort-level with Python now.
5	The likelihood of using Python for future engineering problems.
6	The level of support received from the instructor for using Python.
7	The level of support you will receive from your employer when addressing a problem in your professional career.
8	Your preference for open-ended design problems that do not provide a single answer.
9	The likelihood of professional engineering problems to be more open-ended.

2) Open-ended Question

Besides the rating questions, a single open-ended question was presented to solicit qualitative feedback on the exercise overall. The open-ended question prompt stated, "Share your comments on the Python-Colab project you just completed. If you have not completed the project yet, simply state your current attitude towards this project." The same questionnaire was administered before and after the lab module, again being ungraded and anonymous.

IV. RESULTS AND DISCUSSION

A. Direct: Python Learning Outcome Assessment

The pre-test that assessed Python fundamentals received 100% response rate for both the courses, whereas the post-test assessments received 72% for Spring 2021 and 61% for Summer 2021 course. Fig. 1 presents the pre- and post-test mean scores for both the terms. The pre- and post-tests included 10 basic Python programming related questions. With an increase in the mean scores for the post-tests, the results suggest students gained comfort with Python programming fundamentals as a result of the lab exercise. The greatest improvement was observed for the syntax and command related questions. The large standard deviation for the pre-tests also suggested varying familiarity with Python at the beginning, which narrowed after the lab exercises. The direct assessment outcomes were also reflected in the student perception survey.

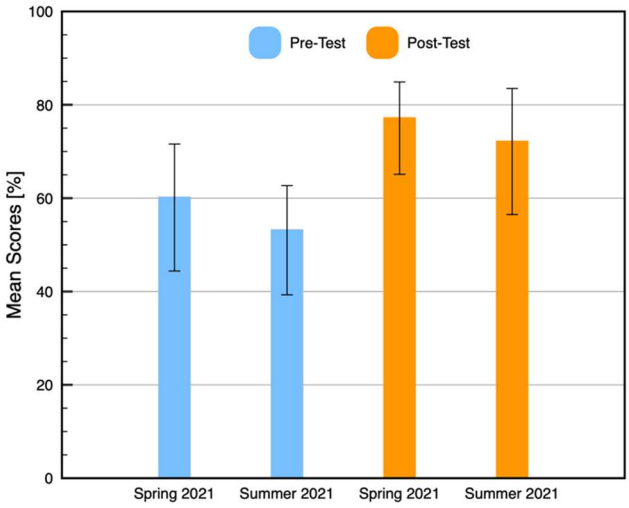

Fig. 1. Mean scores of the pre- and post-tests assessing fundamental knowledge of programming with Python. Error bars represent a single standard deviation.

B. Indirect: Student Perceptions Questionnaire

1) Rating Questions

Fig. 2 presents a summary of the ratings responses received from the students pre- and post-lab exercise. Questions 1 through 4 probed student exposure and comfort with Python. Students exhibited a range of comfort levels with programming (Question 1). As expected, most students did not have prior experience with Python (Question 3). After the lab exercise, both Question 1 demonstrated a positive shift suggesting improved comfort with programming. Specifically, when asked about their comfort level with Python (Question 4) more students rated it 'high' after the lab exercise.

Questions 5 through 9 broadly surveyed student attitudes towards open-ended problems and their perceptions of professional engineering environments. In other words, are students aware of how much independent work they will need to conduct and how open-ended their career projects will be. This reflection was important to reinforce the design choice of this project. Based on Question 9, students are well aware that professional engineering problems will be open-ended. At the same time, Question 8 points to their lower preference for such problems. After the lab exercises, students did not change their preference for open-ended problems by much. In fact, the shifts between the two terms are in the opposite directions. With regards to independent work, responses to Questions 6 and 7

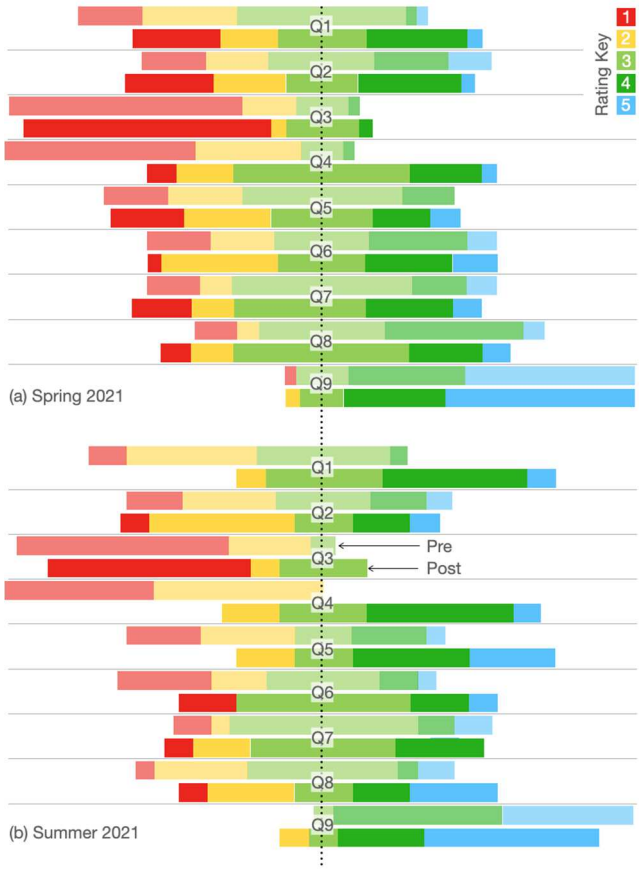

Fig. 2. Likert-Scale rating questions represented for both the (a) Spring 2021 and (b) Summer 2021 courses. The figures compare each pre- and post- responses for each of the nine questions surveyed presented in Table I. The ratings range from 1 for "low" to 5 for "high".

students expect some level of support received from their superiors. Their attitudes towards this did not change noticeably after the lab exercise. Combined with the earlier questions related to open-ended projects, the survey suggests students are used to single-answer (closed-ended) problems that require minimal assistance from their superiors. For open-ended projects that are inherently more challenging, the students expect more guidance from faculty or supervisors. Similar student attitudes towards open-ended problems have been previously noted [22]. The lack of preference for open-ended projects was likely greater for this project considering the highly independent nature of the exercise. Students had to learn Python independently and develop the requisite analysis. With repeated exposure and proper framing of the problem these reservations can be reduced.

Question 5 asked the students about the likelihood of using Python for future engineering problems. There was a notable shift towards higher likelihood after the lab exercise for the Summer 2021 course. The positive shift for Question 5 was minimal for the Spring 2021 term, however. Therefore, it is difficult to conclude if students viewed learning Python programming skills as useful in their careers. We suspect the impact would be greater over a longer time scale once students recognize the prevalence of Python within engineering the profession.

2) Open-Ended Question

Table III provide summaries of the open-ended question analysis. They present a tally of recurring response themes prior to the lab completion and after the lab completion for both terms. The themes are grouped into positive and negative responses.

TABLE III. TABLE OF THEMES GENERATED USING THE OPEN-ENDED PRE- AND POST-LAB SURVEY QUESTION, "SHARE YOUR COMMENTS ON THE PYTHON-COLAB PROJECT YOU JUST COMPLETED. IF YOU HAVE NOT COMPLETED THE PROJECT YET, SIMPLY STATE YOUR CURRENT ATTITUDE TOWARDS THIS PROJECT." THE FREQUENCY OF RESPONSES CORRESPONDING TO THE THEME ARE TALLIED AND GROUPED FOR BOTH SPRING AND SUMMER 2021 COURSES.

Pre-Lab Survey Summary

Positive Feedback			*Negative Feedback*		
Theme	***Spr.***	***Sum.***	***Theme***	***Spr.***	***Sum.***
Excited about the project	7	1	Nervous about coding	5	5
Views as an opportunity to learn a new language	9	7	Nervous about the project in general	5	1
Foresees future utility	6	3	Grading concern	3	2
Expects obstacles yet optimistic	11	2	Time-management concern	3	0

Post-Lab Survey Summary

Positive Feedback			*Negative Feedback*		
Theme	***Spr.***	***Sum.***	***Theme***	***Spr.***	***Sum.***
Useful experience	6	6	Colab/Python issues	3	2
Manageable	6	3	Time consuming	5	2
			High difficulty	9	4
			Physical project preferred	2	0
			Dislike for open-ended project	1	0
			More guidance requested	0	2

The open-ended responses complement the rating responses discussed earlier. Beginning with the pre-lab responses presented in Table III, the students generally have a positive attitude towards the project and view the exercise as an opportunity to learn this new language that they have heard about. They also foresee how this new tool will be helpful in their careers. At the same time, students are concerned about their comfort with programming. While some are confident they can overcome the challenges, a notable few worry that their lack of comfort will hurt their performance, specifically stating their lack of prior experience. Related, is the amount of time required to troubleshoot and if that will in turn have a negative impact on their grades. The following response from the Summer 2021 student captures the student perception prior to the project well.

"I've never used Python before and I have not coded that much in general, so I'm not looking forward to the project, but I think it will be good to have some kind of introduction to the Python language as I think it will be a good tool for the future."

After the lab exercise, students claimed the experience was useful and manageable. At the same time, a few students also stated the project was difficult and required significant investment of time (beyond 20 hrs quoted by the instructor).

Based on this response from the Spring 2021 term, the Summer 2021 project was converted to a paired project. This was also prompted by the fact that the Summer 2021 term was half as long as the Spring 2021. As a result, there is a drop in the frequency of responses for the difficulty and time effort categories. While the relationship between the unpaired and paired project outcomes between the two cohorts was not studied, these are aspects that can influence the learning outcomes and student attitudes [23]. For instance, team-based programming projects often lead to a single person developing the bulk of the code.

To provide a broader context to the discussion of student feedback post-project the following set of representative student responses are presented.

"The Python-Colab project I've completed was on the analysis of various working cycles based on material learned from an Intro to Thermal Fluid Sciences course. The project was less Python heavy than Intro to Thermal Fluid Science material heavy which allowed us to grasp more of a focus on the material we learned in the course while trying a new platform for coding. This project was a good and important experience to grow comfortable in unfamiliar environments using the knowledge we were taught while learning the basics of Python."

"The project was manageable as somebody who has never programmed in Python before. The problems were engaging, and it felt good to produce a visual that seemed to map onto reality."

"Was much harder than expected. At a certain point the physics was lost to the coding and honestly left me far more confused about the physics behind the code. I spent so much time on it that looking at, that it now makes me feel ill."

It is evident from the surveys that students were not comfortable with the new coding requirement even with the structured guidance provided. Their introductory experience with C++, MATLAB, or Java during their first year did little to develop computational confidence outside of computer science courses. This aspect is not surprising since students need practice to develop proficiency [6]. Projects like the one presented here, are ideal for students to develop computational comfort beyond learning the fundamentals. At the same time, when exposing a new programming language, while most students can overcome the initial challenges there will be those who are discouraged unless adequate support is provided [15]. As a result, several gaps in the guidance were identified to assist students in the future. Concurrently, it is important to frame the design of the project accurately and emphasize the utility of computational thinking within engineering. This can enhance the perceived value of the exercise. It is expected that the combined effect of the changes can improve the future implementation of the project.

The project represents an implementation of computational thinking into a single engineering course. However, a systematic integration of computational tools throughout the curriculum can better prepare our students for their careers [6],[8],[17]. A similar implementation using Python has been designed for other courses in the curriculum. Besides, computational thinking the project also allowed students to practice their understanding of the basic thermodynamic concepts. Some students noted that the project helped them prepare for the final exam and also provided them with a different perspective of thermodynamic analysis that can be carried out using computational tools. The thermodynamic learning outcomes, however, were not studied but are expected to have a positive effect.

V. Conclusion

This lab module demonstrates Python's utility in solving advanced thermodynamic analysis, at the same time, equipping our students with this highly desired skill. The lab module is ideally suited for remote instruction but can also be implemented within a non-remote setting. The use of Google Colab notebook facilitated a consistent student experience with a guided framework to promote learning. Both direct and indirect assessments indicate students gained working knowledge of Python. The lab exercise required students to learn Python independently and use it to conduct parametric analysis and evaluation of various thermodynamic cycles. The students recognized the utility of Python in their future careers, yet their preference for open-ended projects was low. The resistance to open-ended projects likely stems from the overreliance on close-ended assessments that are common in academia. Projects like the one presented here can provide a more realistic experience of engineering in the field.

Acknowledgements

The authors thank Dr. Christopher Martin, Dr. Joseph Ranalli, and Dr. Jacob Moore from Pennsylvania State University for developing PYroMat Thermodynamic library that made this laboratory module possible. Their extensive documentation and publications were instrumental in our success.

References

[1] M. Olumolade, "Maximizing student learning through hands on activities experiments in an engineering technology program," Paper presented at 2004 ASEE Annual Conference & Exposition, Salt Lake City, Utah, June 2004.

[2] J. Swenson, M. Rola, A. Johnson, E. Treadway, A. Nitingale, H. Koushyar, J. Lee, and K. Wingate, "Consideration for scaffolding open-ended engineering problems: instructor reflections after three years," 2021 IEEE Frontiers in Education Conference (FIE), pp. 1-8, 2021.

[3] J. Swenson, K. Beranger and A. W. Johnson, "How students take up open-ended, real world problems," 2021 IEEE Frontiers in Education Conference (FIE), pp. 1-5, 2021.

[4] N. Okaniti, "Teaching cycle optimization in introductory thermodynamics courses," 2002 ASEE Annual Conference & Exposition, June 2002.

[5] D. Zietlow, "Optimization of vapor compression cycles" Paper presented at 2014 ASEE Annual Conference & Exposition, June 2014.

[6] E. Wiebe, C. Ho, L. Bullard, D. Raubenheimer, J. Joines, C. Miller, and G. Rouskas, "Computing across the curricula: the view of industry leaders," Paper presented at 2009 ASEE Annual Conference & Exposition, June 2009.

[7] P. Sundaravadivel, "Integrating computational thinking in an interdisciplinary programming course for engineering undergraduates" Paper presented at ASEE 2021 Gulf-Southwest Annual Conference, Waco, Texas, March 2021.

[8] B. Czerkawski, "Instructional design for computational thinking," 2015.

[9] T. Eldredge, and J. H. Jones, "An integrated thermal science MATLAB project," Paper presented at 2020 ASEE Virtual Annual Conference, Virtual Online, June 2020.

[10] T. Dent, K. Woodbury, and R. Taylor, “Microsoft Excel heat transfer add-in for engineering courses,” Paper presented at 2008 Annual Conference & Exposition, Pittsburgh, Pennsylvania, June 2008.

[11] B. R. Bernard, and J. Straub, “Pandemic response: hybrid-flexible course delivery for general education computer science courses,” Paper presented at 2021 ASEE Virtual Annual Conference, Virtual Conference, June 2021.

[12] H. Fangohr, “A comparison of C, MATLAB, and Python as teaching languages in engineering,” International Conference on Computational Science, 2004.

[13] A. Kavianpour, and S. Kavianpour, “The first course of programming: Python, Matlab, or C?” Paper presented at 2016 ASEE Annual Conference & Exposition, June 2016.

[14] E. Ebrahimzadeh and N. Safai, “Should “Python for Engineers” be a course taught to freshmen engineering majors in the U.S.A. and abroad?” Paper presented at 2019 ASEE Annual Conference & Exposition, Tampa, Florida. June 2019.

[15] C. Wang, “A brief introduction of python to freshman engineering students using multimedia applications,” 2020 IEEE Frontiers in Education Conference (FIE), pp. 1-8, 2020.

[16] B. J. Furman, S. Ahsan, and E. Wertz, “Making the move from C to Python with mechanical engineering students,” Paper presented at 2020 ASEE Annual Conference & Exposition, Virtual Conference, June 2020.

[17] M. Wende, T. Giese, S. Bulut and R. Anderl, “Framework of an Active Learning Python Curriculum for First Year Mechanical Engineering Students,” 2020 IEEE Global Engineering Education Conference (EDUCON), pp. 1193-1200, 2020.

[18] C. R. Martin, J. P. Moore and J. A. Ranalli, “Teaching the foundations of thermodynamics with PYro,” 2016 IEEE Frontiers in Education Conference (FIE), pp. 1-6, 2016.

[19] C. R. Martin, J. Ranalli, J.P. Moore, “Problem-based learning module for teaching thermodynamic cycle analysis using PYroMat,” Paper presented at 2017 ASEE Annual Conference & Exposition, 2017.

[20] R. Otis, and Z. K. Liu, “pycalphad: CALPHAD-based computational thermodynamics in Python,” Journal of Open Research Software, 5(1), p.1, 2017.

[21] J. Reisel, *Principles of Engineering Thermodynamics* (1st ed.). Cengage Learning, 2016,

[22] J. L. Hertz, “Confidently uncomfortable: first-year student ambiguity tolerance and self-efficacy on open-ended design problems,” Paper presented at 2018 ASEE Annual Conference & Exposition , Salt Lake City, Utah, June 2018.

[23] P. Korber and R. Motschnig, “The effects of pair-programming in introductory programming courses with visual and text-based languages," 2021 IEEE Frontiers in Education Conference (FIE), pp. 1-9, 2021.