A Brief Introduction of Python to Freshman Engineering Students Using Multimedia Applications

Chao Wang
Ira A. Fulton Schools of Engineering
Arizona State University
Tempe, AZ, USA
chao.wang.6@asu.edu

Abstract— **This Research to Practice Full Paper presents the experience and evaluation of teaching python to freshman engineering students in an Introduction to Engineering course. Most students enrolled in this course are first-semester aerospace, chemical, electrical and mechanical engineering students, with one third (more than half in female students) have zero prior programming experience. A four-week module was developed to teach Python during four 50-minute lecture periods through multimedia applications. Situational Motivation Scale (SIMS) survey, an instrument to measure four types of motivation (intrinsic motivation, identified regulation, external regulation and amotivation) based on Self-Determination Theory (SDT), was administered weekly at the end of the Python lectures. An end-of-semester survey was also given to collect student feedback on the Python module. Survey results reveal that the module achieved the goal of introducing Python programming to freshman engineering students of all majors, with most students exhibiting positive motivational responses to the Python activities due to perceived value, enjoyment and potential future use. However, there were some experiencing less motivation due to a lack of knowledge on programming basics.**

Keywords—First and Second Year Programs, Teach Programming, Python

I. INTRODUCTION

How to equip engineering students with the latest skills to meet the technological demands of the future workforce in an increasingly automated world is a challenge for all engineering educators.

With the rise of automation and artificial intelligence (AI), acquiring basic programming skills is becoming more and more relevant to all engineering students regardless of majors. Yet, programming courses are not always required in engineering disciplines other than electrical and computer engineering. To equip engineering students with this important skill and align with Alan Perlis' vision of teaching programming to *all* students [1], a four-lecture Python module was developed to teach students programming basics in a freshman Introduction to Engineering course, which is required by all engineering majors.

Python is chosen because it is the world's fastest growing programming language. IEEE Spectrum has placed Python in the first spot of the top 10 programming languages since 2017 [2]. It is simple, free and open source. Yet it is also very powerful and has an active community to contribute numerous modules, packages and libraries, which can solve a wide variety of problems across many disciplines. Learning the basics of Python early in the curriculum is useful for engineering students, who can then apply it to their course projects, student club activities, competitions and senior design projects. Even after they graduate, they can keep using Python for work productivity and/or personal do-it-yourself (DIY) aspirations.

Instead of teaching programming in the traditional way, this lesson plan focuses on teaching Python through multimedia applications including sound, image and video. In the context of an image application, a brief introduction to machine learning is also included to expose students to the current state of the art technologies.

This paper will discuss the design and deployment of the four-lecture Python module in four sections of Introduction to Engineering course with nearly 160 students enrolled in the fall semester of 2019. As for assessment, during the four weeks of the Python module, a Situational Motivation Scale (SIMS) survey [3] was given to students each week to measure student motivation. Scores for "intrinsic motivation", "identified regulation", "external regulation" and "amotivation" were calculated, along with a self-determination index [4]. An End-of-Semester survey was also given to students, collecting feedback on students' experience with the Python module.

The rest of the paper is organized as follows. First, prior work on teaching programming through multimedia applications, an overview of the Introduction to Engineering course along with self-determination theory are briefly reviewed in the background section. The module design and implementation are described next, followed by the assessment and results. Then lessons learned and future work are presented, followed by conclusion.

II. BACKGROUND

A. Existing Work

Georgia Institute of Technology developed an introductory media computing course, one that is offered by the College of Computing and required of all incoming Non-CS major students [5]. In the course, students learn to write Python programs to manipulate sound, images and movies.

Assessment showed improved success rate, defined as the percentage of students earning an A, B or C. An online survey also indicated continued engagement with computer science after course ended [6].

Despite both using multimedia applications to teach programming concepts, there are considerable differences between the media computing course and the Python module descried in this paper. The media computing course relies on a custom developed set of tools and APIs, while the Python module in this paper uses popular open source Python IDE and packages. The media computing course is a semester-long course, whereas this Python lesson plan is a four-lecture module that can be plugged in any course to give a brief introduction to programming and Python. Although this Python module does not go as in depth to teach programming as the media computing course due to lack of time, the module does expose students to advanced concepts such as machine learning to stay relevant with rapid advances in technology.

B. *Introduction to Engineering Course*

Introduction to Engineering at Arizona State university is a 2-credit hour course required by all freshman engineering students. Most students take this course during their first semester in college. The course has a 50-minute lecture and two hour and fifty minutes lab each week for 15 weeks. It focuses on introducing students to the broad topics of engineering design process, engineering model and drawing, teamwork, technical communication, project management and entrepreneurial mindset. In addition, the course also teaches technical knowledge such as programming a microcontroller and computer-aided design including 3D printing through two multidisciplinary design projects, i.e., a well-defined project during the first half of the semester and an open-ended project during the second half. The fact that this course is required by all freshman engineering students makes it the perfect place to introduce basic programming concepts.

C. *Self-determination Theory*

Self-determination theory (SDT) [7, 8] postulates that different types of motivations may be described on a continuum ranging from autonomous (internal) to controlled (external) motivations. At one end of the spectrum lies intrinsic motivation, which is a deeply internalized state of engagement based on interest, enjoyment, satisfaction and passion. Amotivation is the other extreme, described by a state of learners finding no value and no desirable outcomes in a learning activity which leads to impersonal or non-intentional action. Between the two extremes are identified regulation and external regulation. Identified regulation is a state of actions due to an internal sense of self and perceived value, importance or usefulness of a task. In contrast, external regulation is a state of compliance with external pressure, prompted by avoidance of punishment or contingent reward.

In any given activity, individuals typically express multiple forms of motivation, not solely autonomous/internalized or controlled/externalized. Therefore, it is beneficial to examine a learner’s motivation across the whole continuum of amotivation, external regulation, identified regulation and intrinsic motivation. Characterizing the motivation spectrum curve into motivational response profile [9-11] can help design course activities to prompt more positive motivational responses from students.

III. DESIGN AND IMPLEMENTATION

Four python lessons are developed to use in four 50-minute lecture periods in the Introduction to Engineering course near the end of the semester. The Python module is created under the assumption that students have already had some C-style programming experience with microcontroller through labs and projects before the start of the module. The lectures are designed to move in a fast pace. It aims to give students an overview of Python and its capabilities, and a brief introduction to programming basics.

The learning objectives of the four modules and the associated multimedia applications are shown in Table I.

TABLE I. LECTURE LEARNING OBJECTIVES

Lecture	Application	Learning Objectives
1: Make Music	Audio	Define and use of scalar- and vector-valued variables. Index into a vector. Plot basic 2D graphs.
2: Create an Image	Image	Define and use 2D arrays. Index into an 2D array. Introduce 3D arrays.
3: Apply Color Filter	Image	Introduce loops and functions.
4: Detect Faces	Image/Video	Introduce machine learning.

To help students prepare for the module, before the first lecture, a homework is assigned. The homework let students go through an online interactive Python tutorial [12] covering the basic concepts such as variables, lists, basic operators, conditions, loops and functions. The homework also requires students to download Python software [13] and install packages [14, 15] needed in the up-coming lectures.

To help students during lectures, demo scripts and exercise code skeletons with majority of the code given are provided on course website for students to download during the four lectures.

A. *Lecture 1: Make Music*

This is the first lecture on Python. The lecture starts with emphasizing the importance of learning how to code. It also states the reason why python is chosen due to its ease to learn, open-source and wide range of applications. It then gives an overview of the four-lecture python module and what students will learn from it. It also points out that the module will only give a bare-bones introduction to programming and Python due to time constraint.

Next, the lecture briefly introduces the physics of sound, followed by a demo of a program playing a music note. In this example, the structure of a python program is explained in detail, such as how to import other python modules, how to initialize variables before using them, and how to call built-in functions. The demo also shows how to plot a sound wave, i.e., a sinusoid function, against a time vector and annotate the plot with labels and title. Vector indexing is introduced by plotting a time segment of the sound wave.

The demo is followed by an exercise in which students define two vectors and produce a plot of the two vector variables.

For students who finish the exercise before the end of class, they are asked to run a demo program to see how vectors get concatenated, i.e., how music notes can get concatenated to a music measure and eventually to a song. An extra credit assignment is given after class to let students write a Python program to play any song or compose their own music.

B. Lecture 2: Create an Image

The focus of this lecture is introducing students to the concept of two-dimensional arrays and array indexing. Image arrays are used since they can be easily visualized.

At the beginning of the lecture, how to use a two-dimensional array to represent a grayscale image is introduced. Next a demo is run to show students how to create a blank image and how to call functions from an external python package, i.e., OpenCV [15], to draw a line, a rectangle, a circle on the blank image and how to write a text on the image. Since location and size arguments need to specify when calling these functions, students practice how to index into a two-dimensional array. Following this demo, students are given an exercise to create a blank image, draw a stick figure and write a text message below it.

After the exercise, the two-dimensional array representation of grayscale images is extended to three-dimensional array to represent color images. A demo is given on how to pass the RGB, i.e., red, green, blue value instead of grayscale intensity value to functions to draw colored line, shape and text. Students are then given the exercise to redraw the stick figure in color.

C. Lecture 3: Apply Color Filter

This lecture is intended to introduce students to loops and functions.

The lecture starts with reviewing how grayscale and color images are represented using 2D and 3D arrays. Then for-loops are introduced. Students are given a couple of multiple-choice questions. For each question, a for-loop code snippet is given, in which the color in one of the RGB channels is manipulated, for example, the blue channel is set to zero, or red channel is reduced by half. Students are given four images and they need to choose the correct image processed by the for-loop. Students are then given a demo on how the for-loop is incorporated in a Python program to process the image. A separate demo is given to show the convenience of setting up a function and passing parameters to the function without rewriting the code.

After the demo, students work on exercises to generate different effects on an image [16] using loops and functions, such as creating a sunset effect, a negative effect or converting a color image to a grayscale image. For students who finish early, they are asked to apply a Sepia filter to an image.

Note for all these filter exercises, filter formulas are given to students. They simply need to code the formulas into the for-loops to apply the effects.

D. Lecture 4: Detect Faces

At the beginning of this lecture, a brief instruction to machine learning and trained classifiers for face detection is given. Students are then given a demo on how to detect faces in an image and draw a box around them. As an exercise, students are asked to detect a cat face in an image using pre-trained classifiers. Lastly, a live demo is given to detect faces in a webcam video in real time.

The purpose of this last lecture is to show students that there are many free open source Python modules and packages on the internet that can perform powerful tasks. As a user, one simply needs to learn how to call and use these existing python modules to solve one's own challenges, and the tasks can often be done using only a few lines of code.

IV. Assessment and Results

The four-lecture Python module was incorporated in four sections of Introduction to Engineering course in the fall semester of 2019. The module was taught near the end of semester in lectures while students were building their open-ended design project in labs. Among the roughly 160 enrolled students, there were 140 students in total who consented to participate in the research study approved by the Institutional Review Board (IRB).

A Situational Motivation Scale (SIMS) survey [3] was given to students at the end of all Python lectures to measure their motivational responses to the lecture activities. Scores for "intrinsic motivation", "identified regulation", "external regulation" and "amotivation" are calculated. A metric called self-determination index (SDI) [4], which is defined as SDI = 2*(intrinsic motivation) + 1*(identified regulation) – 1*(external regulation) – 2*(amotivation), is also calculated. The SDI number is used to represent students' overall motivation by weighing subscale constructs according to their position on the self-determination continuum. The range of possible SDI scores is from -18 to 18, with higher scores indicating greater self-determination.

An end-of-semester survey was given at the end of the course asking student for feedback on the Python module. The questions are Likert-type scale response questions with seven choices: 1 (corresponds not at all), 2 (corresponds very little), 3 (corresponds a little), 4 (corresponds moderately), 5 (corresponds enough), 6 (corresponds a lot), 7 (corresponds exactly).

In both SIMS survey and end-of-semester survey, there is one open-ended free response question asking students if they have anything to share regarding the Python programming activities.

A. SIMS Survey

a) Overall Motivation: There are 303 Situational Motivation Scale (SIMS) survey responses received from 123 students for the four-week Python module. Subscale mean values from the full dataset are shown in Fig.1. The variable M represents the number of unique students contributed to the survey and the variable N denotes the number of unique survey responses. The figure shows, on average, students in this study show low amotivation, moderate external regulation, high identified regulation and relatively high intrinsic motivation. The mean self-determination index is 4.44±0.67 (with possible range from -18 to 18). The shape of the curve in Fig. 1 is in between of two motivational response profiles, i.e., "strongly motivated" and "moderately autonomous" described in [10]. On average, students experience a sense of interest and enjoyment doing the in-class Python programming activities, but also rely on external control and internal regulation to guide their actions.

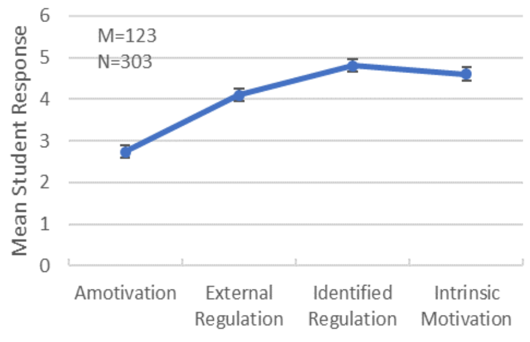

Fig. 1. Situational motivation subscale responses for all students. Error bars show 95% confidence.

b) Motivation by Major: The Introduction to Engineering course enrolls students from different majors in the engineering school including Aerospace Engineering, Chemical Engineering, Electrical Engineering, and Mechanical Engineering. There are occasionally Material Science and Computer Science students enrolled in the course as well. Figure 2 shows the mean self-determination index across all majors.

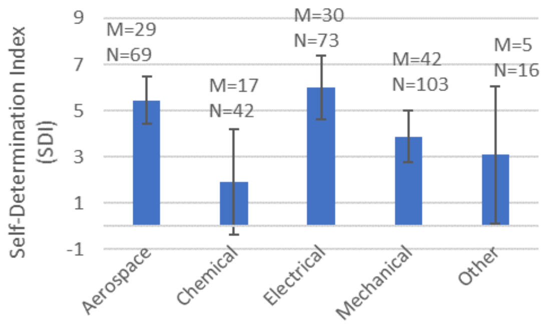

Fig. 2. Mean self-determination index for different majors. Error bars show 95% confidence.

The figure shows electrical engineering students experience the most positive motivation followed by aerospace engineering students. The students from these two majors perceive Python as "useful" and "important". On the other hand, chemical engineering students shows the least motivation because "it is hard and as a chemical engineer (it) makes no sense (and) why I should care." And "I (he/she) wasn't very interested in learning how to use python. I (he/she) would much rather do a different activity that was more related to my (his/her) major." Similarly, "I (he/she) do not enjoy coding, nor am I (he/she) sure how it's going to help me as a chemical engineer."

c) Motivation by Gender: Figure 3 shows the mean self-determination index for male and female students. The figure shows female students exhibit slightly less positive motivational response compared to male students, but the difference is not large, with mean SDI being 4.27 for female students and 4.48 for male students. The lower motivational response can be attributed to the fact that more female students have zero prior programming experience compared to male students. They seem to have more difficulties learning the programming concepts and getting the programs to work. This is manifested by the fact that there are more comments from female than male students related to program not working on their computer, activity being difficult due to no prior coding experience, preference to learning more programming basics.

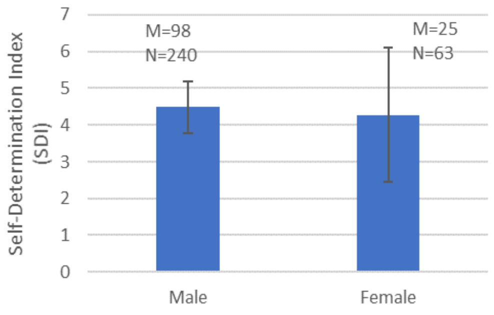

Fig. 3. Mean self-determination index for different genders. Error bars show 95% confidence.

d) Motivation by Ethnicity: Figure 4 shows the mean self-determination index for different ethnicities. Non-white students include roughly 39% Hispanic, 27% Asian, 8% Black, 5% Native American and 21% others.

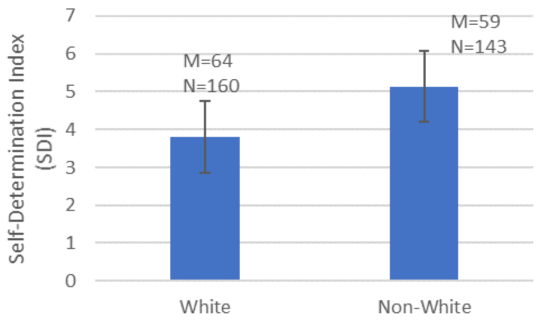

Fig. 4. Mean self-determination index for different ethnicities. Error bars show 95% confidence.

Figure 4 shows non-white students demonstrate higher self-determination compared to white students, with mean SDI being 5.14 for non-white students and 3.81 for white students. The mean scores of situational motivation subscale responses show that with amotivation and external regulation almost the same for white and non-white students, non-white students exhibit more identified regulation (4.98 vs. 4.65) and intrinsic motivation (4.83 vs. 4.41). In other words, non-white students find more interest, enjoyment, importance, and usefulness in the activities than white students.

B. *End-of-Semester Survey*

There are 98 students who completed the End-of-Semester survey, which includes five questions on the Python module:

a) Prior Programming Experience: Figure 5 shows students' responses to survey question "I have no programming experience prior to this class." Around one third of all students responded have no programming experience, i.e., response 7. Even though not showing in the figure, as high as 53% of female students have zero prior programming experience.

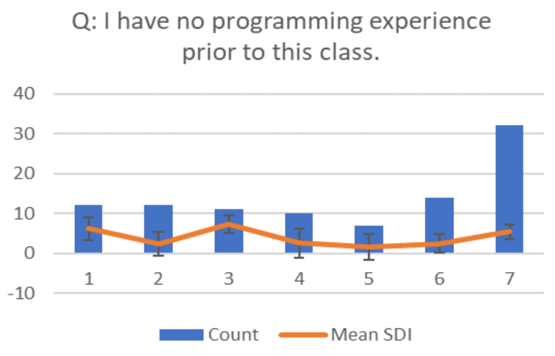

Fig. 5. Student count histogram of survey responses to "I have no programming experience prior to this class." Choices are 1: corresponds not at all, 2: corresponds very little, 3: corresponds a little, 4: corresponds moderately, 5: corresponds enough, 6: corresponds a lot, 7: correspond exactly. Mean SDI for each response is also plotted. Error bars show 95% confidence.

The mean SDI score for each student bin with responses 1 through 7 is also plotted on Fig. 5. No clear pattern is found. The top three SDI scores are for responses 3, 1, and 7, with mean SDI score of 7.33, 6.16, 5.36. It is worth noting that having no prior programming experience does not translate to low motivation, just as some student commented "I (he/she) know(s) nothing about coding but I (he/she) would like to learn."

b) Learning Programming Basics: Figure 6 shows students' responses to survey question "The Python in-class activities help me learn the programming basics." 71% of all students feel that the Python activities help them learn the programming basics at least moderately, i.e., responses 4 and higher.

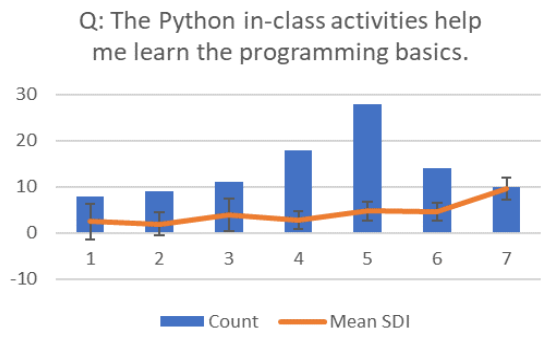

Fig. 6. Student count histogram of survey responses to "The Python in-class activities help me learn the programming basics." Choices are 1: corresponds not at all, 2: corresponds very little, 3: corresponds a little, 4: corresponds moderately, 5: corresponds enough, 6: corresponds a lot, 7: correspond exactly. Mean SDI for each response is also plotted. Error bars show 95% confidence.

The mean SDI curve on Fig. 6 shows ups and downs. One thing to note is that the students in bin 7 have the highest mean SDI score (9.60), i.e., the perception of learning a new skill, i.e., programming basics, is positively correlated to higher motivation.

c) Difficulty Level: Figure 7 shows students' responses to survey question "The Python in-class activities are at the appropriate difficulty level." Again 71% of all students respond at least moderately, i.e., responses 4 and higher.

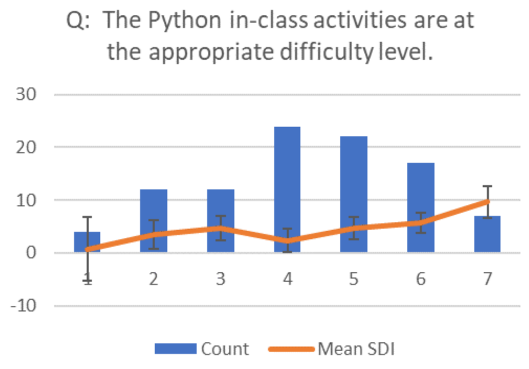

Fig. 7. Student count histogram of survey responses to "The Python in-class activities are at the appropriate difficulty level." Choices are 1: corresponds not at all, 2: corresponds very little, 3: corresponds a little, 4: corresponds moderately, 5: corresponds enough, 6: corresponds a lot, 7: correspond exactly. Mean SDI for each response is also plotted. Error bars show 95% confidence.

The two extremes on the mean SDI curve in Fig. 7 are worth noting. The students who think the activities are not at the appropriate difficulty level, i.e. response of 1, have the lowest mean SDI score of 0.75. On the contrary, students giving response of 7, i.e., they feel the activities are at the appropriate difficulty level, have the highest mean SDI score of 9.65. This can be explained by self-determination theory [17] which believes the satisfaction of the basic need of competence, i.e., a sense of mastery and self-efficacy, plays an important role in shaping up more positive motivational response. In this case, when a student feels the activities are too difficult, he/she is not motivated to learn at all.

d) Fun Applications: Figure 8 shows students' responses to survey question "I feel like learning Python through a multimedia approach, i.e., audio, image, video is fun." 70% of all students respond at least moderately, i.e., responses 4 and higher.

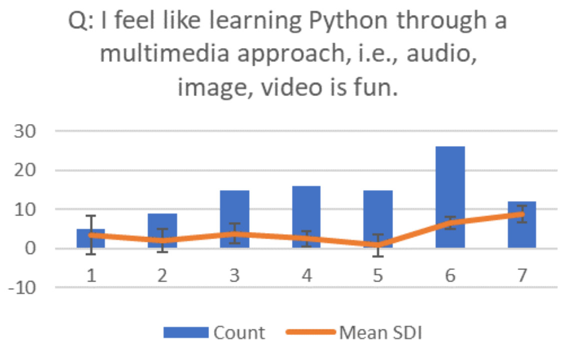

Fig. 8. Student count histogram of survey responses to "I feel like learning Python through a multimedia approach, i.e., audio, image, video is fun." Choices are 1: corresponds not at all, 2: corresponds very little, 3: corresponds a little, 4: corresponds moderately, 5: corresponds enough, 6: corresponds a lot, 7: correspond exactly. Mean SDI for each response is also plotted. Error bars show 95% confidence.

For this question, students in bin 7 have again the highest mean SDI score (8.79), i.e., a sense of interest and enjoyment is positively correlated to higher motivation.

e) Future Use: Figure 9 shows students' responses to survey question "I will use Python in the future as the need arises." 73% of all students respond at least moderately, i.e., responses 4 and higher.

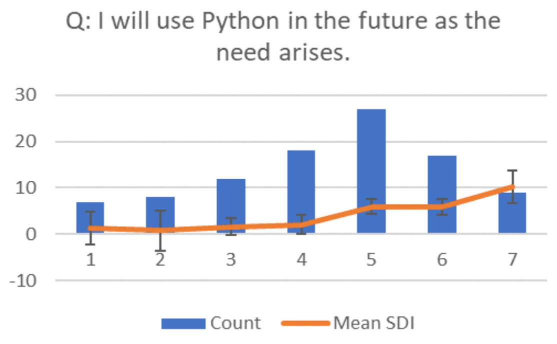

Fig. 9. Student count histogram of survey responses to "I will use Python in the future as the need arises." Choices are 1: corresponds not at all, 2: corresponds very little, 3: corresponds a little, 4: corresponds moderately, 5: corresponds enough, 6: corresponds a lot, 7: correspond exactly. Mean SDI for each response is also plotted. Error bars show 95% confidence.

There is almost a linear relationship between mean SDI and if students plan to use Python in the future. The students responding 7, who will for sure use Python in the future as the need arises, have a mean SDI score as high as 10.19, because "(it is) useful in the future."

C. *Student Free-Form Responses*

Besides the Likert-Scale SIMS and the end-of-semester survey questions, there is also a free response question in each survey to capture anything students want to share.

These free-form responses show that overall students enjoyed the Python programming activities. Below are some examples showing different aspects:

Regarding learning to code and its importance:

- "It was fun to figure out how to code!"
- "Python is very interesting to learn and it is cool to learn the basics of how coding actually works."
- "It is important for people to know coding languages as we are a technologically developed society, so I see great value in it."
- "Overall, this experience has given me an introduction into coding that I feel is essential to an engineering career field."
- "This experience has taught me basic coding skills that I think will be greatly beneficial to my future as an engineer. It will help me throughout my other classes and potentially assist me through an internship."
- "These activities are going to help me in the future in more difficult coding classes, so I appreciate the introduction."

Regarding Python compared to other programming languages:

- "I enjoy that Python is incorporated into this class since it is becoming more popular. I believe that it is beneficial to learn the basics of it."
- "I really enjoyed Python and I was glad we could learn this programming language."
- "The relatively easy ability to get certain tasks done in python."
- "Amazing what a few lines of code can achieve."
- "The experience that sticks out the most is the wide variety of Python to do things, and how much easier it is to use when compared to other programming languages such as C++."
- "Python is amazing, and I look forward to using it in the future."

Regarding using multimedia applications to introduce programming and Python:

- "I thoroughly enjoyed learning how sound frequency works, as well as how to convey that into machine language in order to successfully compile the program through python and play sounds through my computer!"
- "I thought the music was a very interesting change and a cool introduction to coding."

- "I enjoyed learning how to code both sounds and images as I find the possibilities of what can be created there limitless."
- "This is cool learning to use a computer program to create pictures."
- "It is fun changing the colors."
- "This (face detection activity) reminds me of the time when I open my camera app on my phone, and it does the face recognition which is something I thought was really cool."
- "Found the face tracking feature interesting."

On the other hand, there are comments complaining the difficulty of the exercises due to lack of prior programming experience, such as

- "It was hard to learn how to use python with no experience in coding."
- "It was just like jumping into it and if you didn't have any prior coding experience then it was just boring."
- "It was hard for me to learn because we never really went over how the computer thinks. I am new to coding, so I am a little lost."
- "Just struggled learning the basics."

Even though there was a homework assignment for students to install Python and related packages, there were students either didn't do it or couldn't get it to work:

- "Python did not work properly on my laptop which prevented me from doing the activities."
- "My program wasn't working correctly, so I wrote the code, but didn't get too much from the experience."
- "Having a Mac is hard when working with Python."
- "My computer was unable to handle the Spyder program for some reason."
- "I don't really know what we went over in class as the entire time I was trying to figure out how to install the additional python packages, which was kind of frustrating."

Despite these difficulties, students still show willingness to learn due to their perceived value in coding and Python:

- "This activity was difficult just because I know nothing about coding, but I would like to learn."
- "I really do not understand coding, however I am trying my hardest to learn."
- "Python and coding in general are very difficult but extremely rewarding."
- "Python was relatively difficult but important."
- "I feel Python will be useful, I just feel I need a little more practice."

V. Lessons Learned and Future Work

From survey results and student feedback, most perceived the importance of learning a programming language such as Python. They also enjoyed the multimedia application activities and felt the module helped them learn the programming basics which they could use in time to come. These demonstrate that the Python module achieves the goal of introducing Python programming to freshman students in all engineering majors, equipping them with a practical skill that can benefit them in the future. In fact, the four-lecture Python module designed for the Introduction to Engineering course can be used as a standalone module in any course that needs a brief introduction to Python.

However, the lack of coding experience in the student population was underestimated. The lessons were planned based on the assumption that students should have basic knowledge in coding from the microcontroller programming project students already finished in the same course in the first half of the semester. Possible explanation could be that the programming project is team-based and not everyone gets to practice programming. As a result, the fast pace of the module lead to frustration especially among those with zero coding experience.

Some future strategies for improving the four-lecture Python module and help students, especially those with no prior programming experience, get the most out of the module include:

- Teach the basics. Use additional lecture time or pre-recorded videos to go over coding basics, such as basic syntax and construction of a programing language, how a computer program is executed in order or out of order due to conditional and loop statements and function calls.
- Make sure students come prepared. Before the first lecture, use pre-lecture quiz to test if students go over tutorial or videos explaining the basics. Let students submit screenshots before class showing the Python software and required Python packages get installed. Provide out-of-class teaching assistant (TA) and instructor support if help is needed.
- Provide more in-class support. Additional TA support during Python lectures to help answer student questions is very important. Students easily get frustrated when they encounter computer problems or syntax errors that they don't know how to fix.
- Slow down a bit. Some of the in-class exercises can be moved to out-of-class homework. Give students more time in class to run code and ask questions.
- Hold students accountable. Give Python homework and additional exercises to help students practice. Incorporate coding questions into exam.

VI. Conclusion

A four-lecture module introducing Python programming using multimedia applications was piloted in a first-year

Introduction to Engineering course enrolling students from different engineering majors. Survey results showed that most students exhibited positive motivational responses during the learning activities and enjoyed the learning experience through multimedia applications due to perceived value, interest and potential future use. These survey results confirmed the value of the Python module as a fun, useful introduction to the programming language. However, students with no prior coding experience struggled through the exercises. To help these students, future improvement on the module includes teaching the basics, slowing down the lectures, providing more support, making sure students come prepared and holding them accountable.

References

[1] M. Greenberger, "Computers and the World of the Future," Transcribed recordings of lectures held at the Sloan School of Business Administration, April, 1961. MIT Press, Cambridge, MA, 1962.

[2] IEEE Spectrum, "The 2018 Top Programming Languages," Retrieved from https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

[3] F. Guay, R. J. Vallerand, and C. Blanchard, "On the Assessment of Situational Intrinsic and Extrinsic Motivation: The Situational Motivation Scale (SIMS)," *Motivation and Emotion*, vol. 24, no. 3, pp. 175-213, 2000.

[4] R. J. Vallerand, "A hierarchical model of intrinsic and extrinsic motivation for sport and physical activity," In M. S. Hagger & N. L. D. Chatzisarantis (Eds.), Intrinsic motivation and self-determination in exercise and sport (pp. 255–279,356–363), Human Kinetics, 2007.

[5] M. Guzdial, "A Media Computation Course for Non-Majors," in Proc. ITiCSE'03, June 30–July 2, 2003, Thessaloniki, Greece.

[6] M. Guzdial, A. Forte, "Design Process for a Non-majors Computing Course," in Proc. SIGCSE'05, February 23-27, 2005, St. Louis, MO

[7] R. M. Ryan, and E. L. Deci, "Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being," *American Psychologist*, vol. 55, no. 1, pp. 68-78, 2000.

[8] J. D. Stolk, Y. V. Zastavker, and M. Gross, "Gender, Motivation, and Pedagogy in the STEM Classroom: A Quantitative Characterization," in Proc. 125st ASEE Annual Conference & Exposition, Salt Lake City, UT, June, 2018.

[9] M. Vansteenkiste, E. Sierens, B. Soenens, K. Luyckx, and W. Lens, "Motivational profiles from a self-determination perspective: The quality of motivation matters," Journal of Educational Psychology, vol. 101, no. 3, pp. 671-688, 2009.

[10] N. Gillet, A. J. S. Morin, and J. Reeve, "Stability, change, and implications of students' motivation profiles: A latent transition analysis," Contemporary Educational Psychology, vol. 51, pp. 222-239, 2017

[11] C. F. Ratelle, F. Guay, R. J. Vallerand, S. Larose, and C. Senécal, "Autonomous, controlled, and amotivated types of academic motivation: A person-oriented analysis," Journal of Educational Psychology, vol. 99, no. 4, pp. 734-746, 2007.

[12] Interactive Python Tutorial. https://www.learnpython.org/

[13] Anaconda, The World's Most Popular Python/R Data Science Platform. https://www.anaconda.com/distribution/

[14] Python-SoundDevice. https://python-sounddevice.readthedocs.io/en/0.3.14/

[15] OpenCV-Python. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

[16] M. Guzdial, B. Ericson, Introduction to Computing and Programming in Python, a Multimedia Approach. Fourth Edition. Pearson, 2016.

[17] E. L. Deci, R. J. Vallerand, L. G. Pelletier, and R. M. Ryan, "Motivation and Education: The Self-Determination Perspective," Educational Psychologist, vol. 26 no. 3&4, pp. 325-346, 1991.